SOMMAIRE

Test Logiciel

SOUKEINA BEN CHIKHA

INTRODUCTION

- Le test est une activité importante dont le but est d’arriver à un produit avec « zéro défaut ».
- C'est la limite idéaliste vers laquelle on tend pour la qualité du logiciel.
- Généralement 40% du budget global est consacrée à l’effort de test.

Motivation

- Impossibilté théorique de prouver la correction de tout programme.
- On teste le programme pour augmenter le niveau de confiance en ce programme

FONDEMENT DU TEST

- Le test est une recherche d'anomalie(s) dans le comportement du logiciel.
- C’est une activité paradoxale : il vaut mieux que ce ne soit pas la même personne qui développe et qui teste le logiciel.
- Un **bon** test est celui qui met à jour une erreur (non encore rencontrée).
- La difficulté réside dans lefait qu'il faut arriver à gérer une suite de tests la **plus complète** possible à un **coup minimal**.
- Un test ne peut pas dire « il n'y a pas d'erreur ».

DÉFINITIONS

- Un programme est vu comme une fonction de transformation d'un ensemble en entrée E vers un ensemble en sortie S.

- Un jeu de test est la définition de plusieurs données de test (DT). Une DT est un élément de E.

- Informations de base nécessaires :

  Programme + spécification de ce programme =Oracle.

DÉROULEMENT

- Préparation de la configuration : Un test se fait selon un sénario de **configuration.**
- L'exécution d'un programme avec comme entrée un jeu de test
- Collecte des résultats produits.
- {x,y, z } après exécution donne {x',y',z'}, il faudra décider si c'est correct ou pas

DOMAINE D'ENTRÉE E

- |E| peut être infini
  - Exemple :
    - E= {ensemble des chaines de caractère}

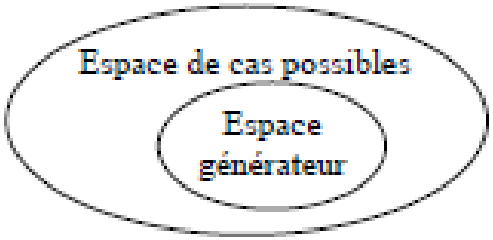
```
pog
ch : cc
début
        Lire (ch);
        Écrire (ch)
Fin
```

    - R= IR = { ensemble des réels}
- Comme l'espace des cas possibles est grand (ou infini) on doit trouver l'échantillon représentatif, puisque on ne peut pas générer un jeu de test avec comme entrée E.

→Tester de façon **exhaustive** un programme est **impossible**

DOMAINE D'ENTRÉE E

- Il faut sélectionner des échantillons représentatifs de E choisis judicieusement.



- Propriétés recherchées : Si l'espace générateur est couvert alors la probabilité d'une défaillance dans l'espace de cas possible est très faible.
- La difficulté est de faire en sorte que l'espace générateur soit consistant et complet.

SPÉCIFICATION D'UN JEU DE TEST

- Un JT est réussi (JT ={t}) pour un programme P accompagné d'une spécification S lorsque $\forall$ t $\in$T, P(t)=S(t).
- Un JT est non réussi si $\exists$ t $\in$T / P(t)$\neq$S(t).
- Exemple :

P

```
début
    lire (n)
    pour i de 1 à n faire
        Res <- Res*i
    FinPour
    ecrire (Res)
Fin
```

SPÉCIFICATION D'UN JEU DE TEST

- E= IN
- S= {x∈IN, y ∈IN/ y=1*2* ...*x}
- Exp de DT = 3 → Res= 449 car Res n'a pas été initialisé

Critère de sélection d'un JT:

Statégie de choix d'un sous ensemble de E pour constituer un JT

Exemple :

S : Programme qui lit en entrée un entier n et qui fournit le nombre premier numéro n (sauf 2).

CLASSIFICATION DES TECHNIQUES DE TEST

- Le testeur choisit une technique de test selon de quoi il dispose pour créer le test.
  - Spécification (ce que fait le programme)
  - Code source (comment est fait le programme)
  - Exécutable (binaire)
- Le testeur peut disposer de 1 , 2 ou des 3 et selon, on aura différentes familles de test.

TEST LOGICIELS

Plus généralement,

- si on dispose du code source, on parle de techniques de tests "**Boites Blanches**" ou "**test structurels**".

- si on **ne** dispose **pas** du code source, on parle de techniques de tests "B**oites Boires**" ou "**test fonctionnels**".

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

- Supposons qu'on dispose de :
  - La spécification
  - L'exécutable.
- Principe :

1. On considère le programme dans son aspect fonctionnel et non plus structurel.
2. On partitionne le domaine (DE) en classes en s'appuyant sur la spécification.
3. On génère des cas de test aux limites de classe.

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

- **Exemple :** partitionnement de E

Spécification: f(x) = $\sqrt{\frac{1}{x}}$

E= IR

C1 : x ≠0

C2 : x=0

C3 : x<0

C4 : x>0

Condition → $\cup C_i = D$, $\forall i, j\ Ci \cap Cj = \varnothing$

Or C1 ∩C3 ≠ ∅ .

→ On en lève C1

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

- On constate que les erreurs se situent très souvent sur les valeurs limite du domaine d'entrée.
- Dans l'approche test aux limites, on suppose qu'on dispose d'une relation d'ordre sur le domaine d'entrée.
- On génère les DT en recherchant les minimas et les maximas de la relation d'ordre sur les classes définies lors de la partition du domaine d'entrées en classes représentatrices.
- On ne teste pas uniquement les minimas et les maximas, on prévoit de tester aussi leurs voisinage ou toute autre valeur qui pourrait sembler pertinante.

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

Exemple 1: tests aux limites

Le domaines en entrée E est partitionné en 3 classes :

- <10 000
- Dans [10 000 , 100 000 [
- > 100 000

Les cas de test aux limites de classes sont donc 00 000 et 09 999 pour la première classe, 10 000 et 99 999 pour la deuxième classe et 100 000 pour la troisième.

- On a donc à tester les nombres suivants :

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

- **Exemple 2 :** tests aux limites.

E = [1, 100]

DT= { 0, 1, 2 , 99,100,101}

- Si le domaine en entrée **E=[Binf., Bsup.]** avec uns pas △.

DT= { Binf.- △,Binf, Binf+ △, Bsup.- △, Bsup., Bsup.+ △}

- Si le domaine en entrée est une **énumération**

DT= 3premières valeurs et 3 dernières valeurs

! Il y a toujours une relation d'ordre.

APPROCHE FONCTIONNELLE OU « BOITE NOIRE »

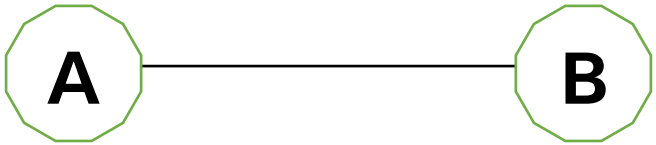
Graphes Causes Effets (cat.5)

- Ces graphes établissent les liens entre les causes (Entrées) et les effets (Sorties) dans un programme.

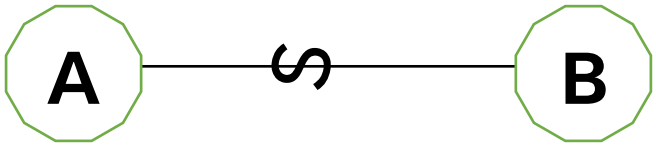
- Quatre symboles :
  - Identité
  - Négation
  - Conjonction
  - Disjonction

GRAPHES CAUSES EFFETS

- Identité : la cause est identique à l'effet.

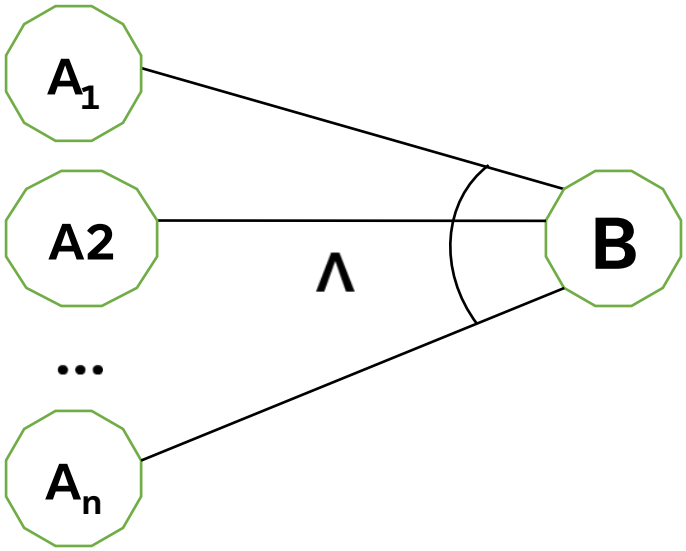


- Négation : en présence de la cause A l'effet B ne peut jamais avoir lieu.



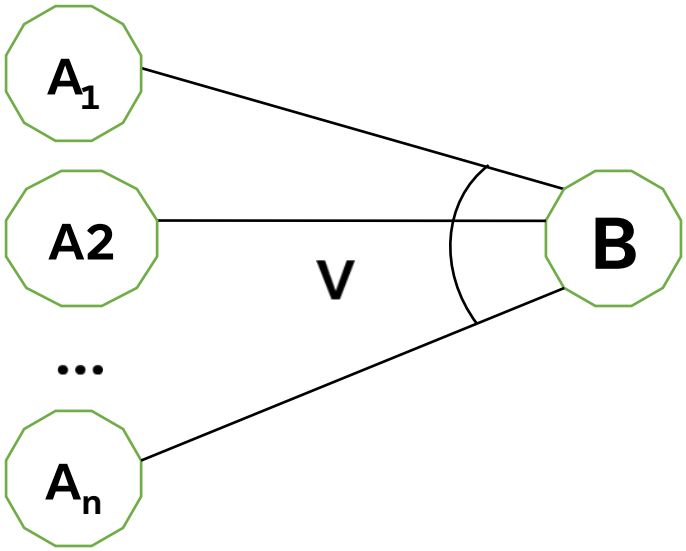
GRAPHES CAUSES EFFETS

- Conjonction : lorsque les effets A1, A2, ..., An sont réunis, l'effet B se produit.



GRAPHES CAUSES EFFETS

- Disjonction : Dès que l'un des effets Ai, se produit, l’effet B se produit.



GRAPHES CAUSES EFFETS

- Exemple :
- Programme de gestion des fichiers. Les fichiers en entrée ont la structure suivante:
  - Le caractère de la première colonne est 'A' ou 'B'
  - Le caractère de la 2ème colonne est un chiffre.

  →Dabs ce cas on ferme le fichier.
  - Si le caractère en première colonne n'est ni 'A' ni 'B' alors on affiche le message d'erreur E1.
  - Si le caractère la 2ème colonne n'est pas un chiffree alors on affiche le message d'erreur E2.

GRAPHES CAUSES EFFETS

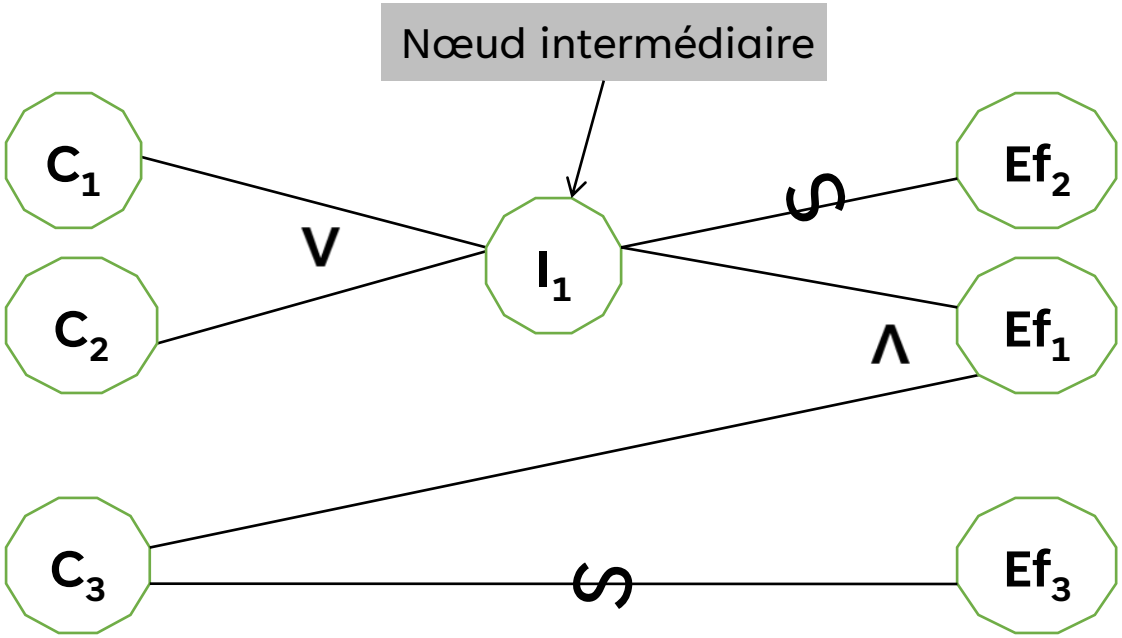
Causes

- C1: carac. en 1ère colonne A.
- C2: carac. en 1ère colonne B.
- C3: carac. en 2ème colonne est un chiffre

Effets

- Ef1: Fichier fermé
- Ef2: Affichage du message d'erreur E1
- Ef3: Affichage du message d'erreur E2

GRAPHES CAUSES EFFETS



GRAPHES CAUSES EFFETS

- Table de vérité avec toutes les combinaisons possibles des causes et effets.

Causes			Effets		
C1	C2	C3	Ef1	Ef2	Ef3
0	0	0	0	1	1
0	0	1	0	1	0
0	1	0	0	0	1
0	1	1	1	0	0
1	0	0	0	0	1
1	0	1	1	0	0
1	1	0	0	0	1
1	1	1	1	0	0

Impossible : on ne peut pas à la fois trouver en première colonne les caractères A et B

GRAPHES CAUSES EFFETS

- On aura 6 jeux de test.

DT={ (CA), (e3) , (BA), (B4), (AT),(A2)}

(CA)→[E1, non fermer fichier, E2]

....

- Une partie du jeu de test est automatisable : vérification de l'echec ou de la réussite du test.

- Cette automatisation a le problème suivant:
  - Les tables de vérités deviennent complexes : explosion combinatoires (exponontielle)

→ Règles de simplification

APPROCHES STRUCTURELLE OU « BOITE BLANCHE »

- Elles reposent sur la représentation du code source par un graphe de contrôle

- Une DT sensibilise un chemin dans le graphe : l'exécution du programme avec cette DT se traduit par un chemin dans le graphe. (Exemple 1)

- Il existe des graphes qui possèdent des chemins non sensibilisables. (Exemple 2)

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »

- Exemple 1:

```
Si  x<=0 alors          (1)
 x<- x-1                (2)
Sinon
 x<- 1-x                (3)
FinSI                   (4)
Si x=-1 alors           (5)
 x<- x-1                (6)
Sinon
 x<- x+1                (7)
FinSi                   (8)
```

- DT={x=2} → 1 3 4 5 6 8

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »

- Exemple 2: chemin non sensibilisable.

```
Si  x≤0  alors  (1)
 x<- -x (2)
Sinon
 si x=0 alors (3)
    x<- x+1 (4)
 sinon
    x<-x-2 (5)
 FinSI (6)
FinSI  (7)
```

- On ne peut jamais passer par le nœud ?.

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »

- Ces approches consistent à analyser la structure interne du programme en déterminant les chemins minimaux.

- Afin d'assurer que:
  - Toutes les conditions d'arrêt de boucle ont été vérifiées.
  - Toutes les branches d'une instruction conditionnelle ont été testés.

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »

- Les différentes techniques de tests Boite Blanche son classées par rapport aux chemins qu'elles essayent de sensibiliser.
  - Couverture de tous les nœuds
  - Couverture de tous les chemins indépendants
  - Couverture de tous les arcs.

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES NŒUDS.

Couverture de tous les nœuds.

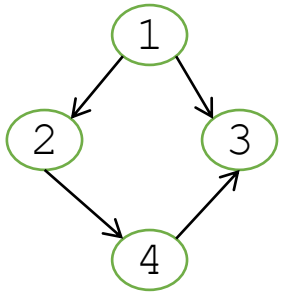
- Générer une DT telle que l'union des chemins sensibilisés donnent tous les nœuds **instructions** du graphe.

Exemple :

```
void Somme (int x, int y)
{
  if( x==0)
        return (x);
  else
        return(x+y);
  //finSi
}
```

1
2
3
4

On a 3 nœuds instructions
DT={x=0, x=5}



APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES NŒUDS.

- On peut claculer l' *"importance"* d'un jeu de tets
- TER *Test Effectiveness Ratio*
- *TER=*

$$\frac{NombreDeNoeudsInstruction \in Ch.UtilidséParlaDT}{NbrDeNoeudsInstruction}$$

APPROCHE STRUCTURELLE OU « BOITE BLANCHE » COUVERTURE DE TOUS LES NŒUDS.

- Exemple :

Lire (x)

Si (x≠0) alors

  x<-1

FinSI

y<- $\frac{1}{x}$

DT={x=4}

NB La technique couvre tous les nœuds, or, elle ne se rend pas compte de l'erreur si x=0

→ Limite de cette technique

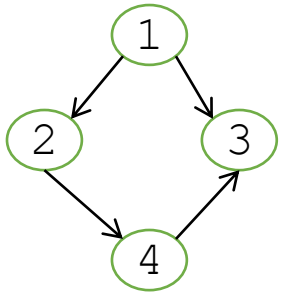
APPROCHE STRUCTURELLE OU « BOITE BLANCHE »

COUVERTURE DE TOUS LES NŒUDS.

Exemple :

```
void Somme (int x, int y)
{
  if( x==0)        1
        return (x);      2
  else
        return(x+y);     3
  //finSi          4
}
```

On a 3 nœuds instructions
DT={x=0, x=5}



DT={ x=0} TER=2/3 =0.66

Interpretation : 66% du travail effectué, il en reste 33%

APPROCHE STRUCTURELLE OU « BOITE BLANCHE » COUVERTURE DE TOUS LES ARCS.

- Il faut générer une DT telle que chacun des arcs du graphe de contrôle est sensibilisé au moins une fois.
- Exemple 1:

lire (x)

Si (x≠0) alors

x<- 1

FinSI

y<- $\frac{1}{x}$

DT={x=4, x=0 }. Le programme échou et on détecte l'erreur.

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES ARCS.

- Exemple 2: Limite

```
Lire (x)
Somme<-0
i<-1
TantQue i<=n faire
  Somme<-Somme+T[i]
  i<-i+1
FinTQ
Ecrire (1/Somme)
```

DT={ n=2, T|3|4|}

TER=1/1 =100%

Dans le cas où n=0 on a une erreur et ce critère ne trouve pas cette erreur.

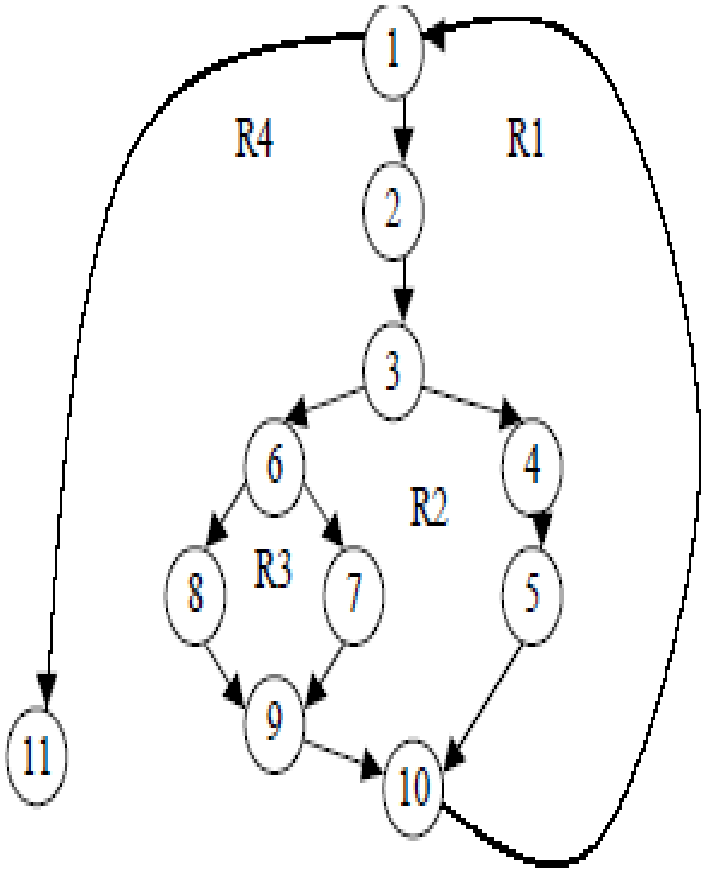
APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES CHEMINS INDÉPENDANTS

- **Mesure de complexité de Mac Cabr.**
  - Cette mesure donne le nombre de chemins minimaux (ou indépendants).
  - Elle est donnée par :

    Nombre cyclomatique =Nb.Arcs – Nb.Noeuds + 2
- Les chemins indépendants d'un graphe de contrôle est l'ensemble des chemins lels que tous les chemins du graphe peuvent s'exprimer par une combinaison linéaire de ceux-ci.

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES CHEMINS INDÉPENDANTS

- On en déduit le graphe de flot suivant.
- Donc le nombre cyclomatique est : Nb.Arcs – Nb.Noeuds + 2 = 13 – 11 + 2 = **4**
- Pour vérifier, on regarde les chemins minimaux (un test par chemin pour tester toutes les possibilités du programme) :
  - a. 1.11
  - b. 1.2.3.4.5.10.1.11
  - c. 1.2.3.6.7.9.10.1.11
  - d. 1.2.3.6.8.9.10.1.11



APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES CHEMINS INDÉPENDANTS

- Principe :

Répéter

chercher premier arc non marqué.

générer le chemin le plus court à partir de cet arc jusqu'à la sortie et marquer les arcs.

jusqu'à (tous les arcs soient marqués)

- TER=

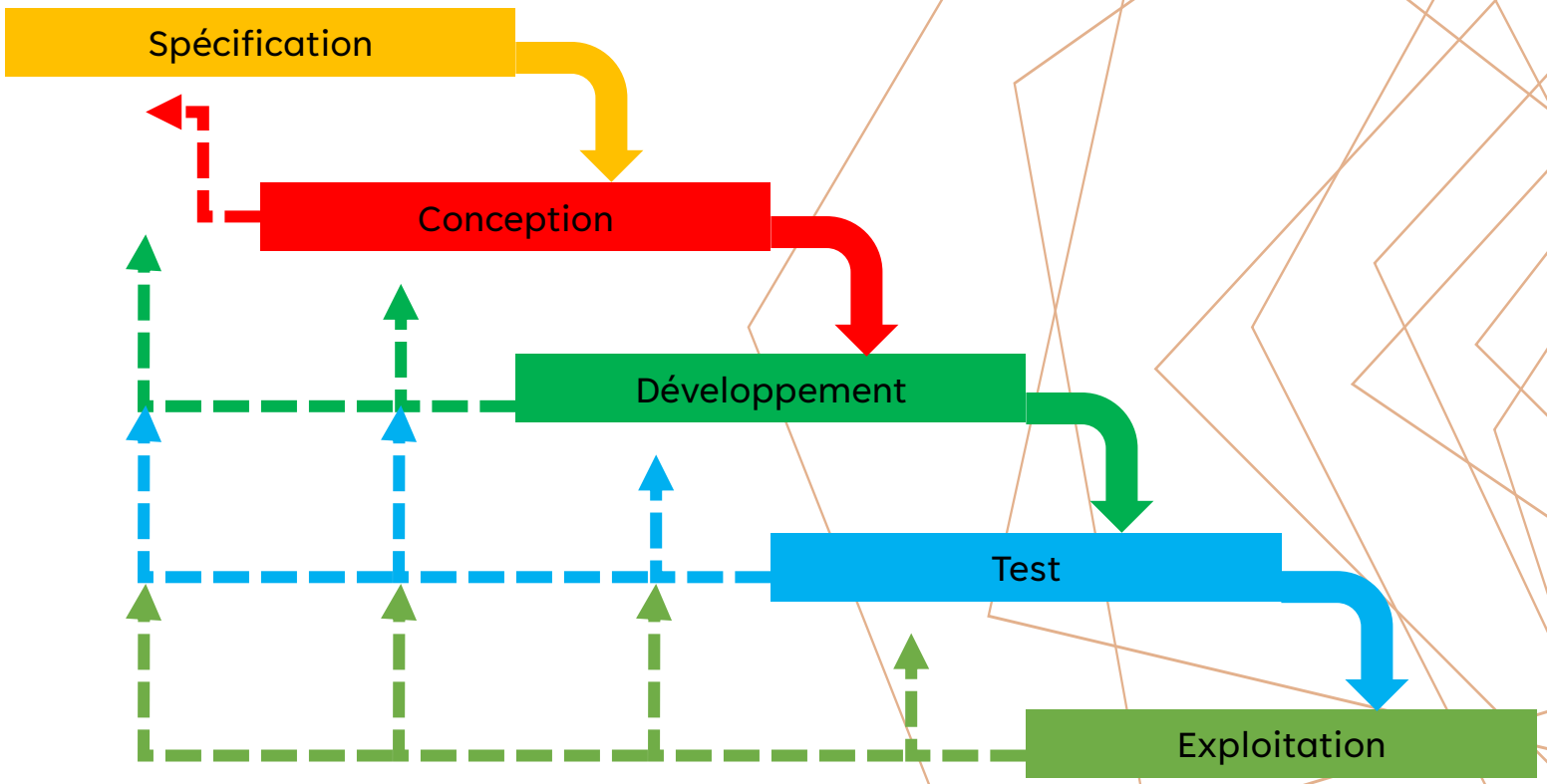
$$\frac{NombreDeChemiinsSensibilisés}{NbrD'arcs - NbrNoeud + 2}$$

APPROCHE STRUCTURELLE OU « BOITE BLANCHE »
COUVERTURE DE TOUS LES CHEMINS INDÉPENDANTS

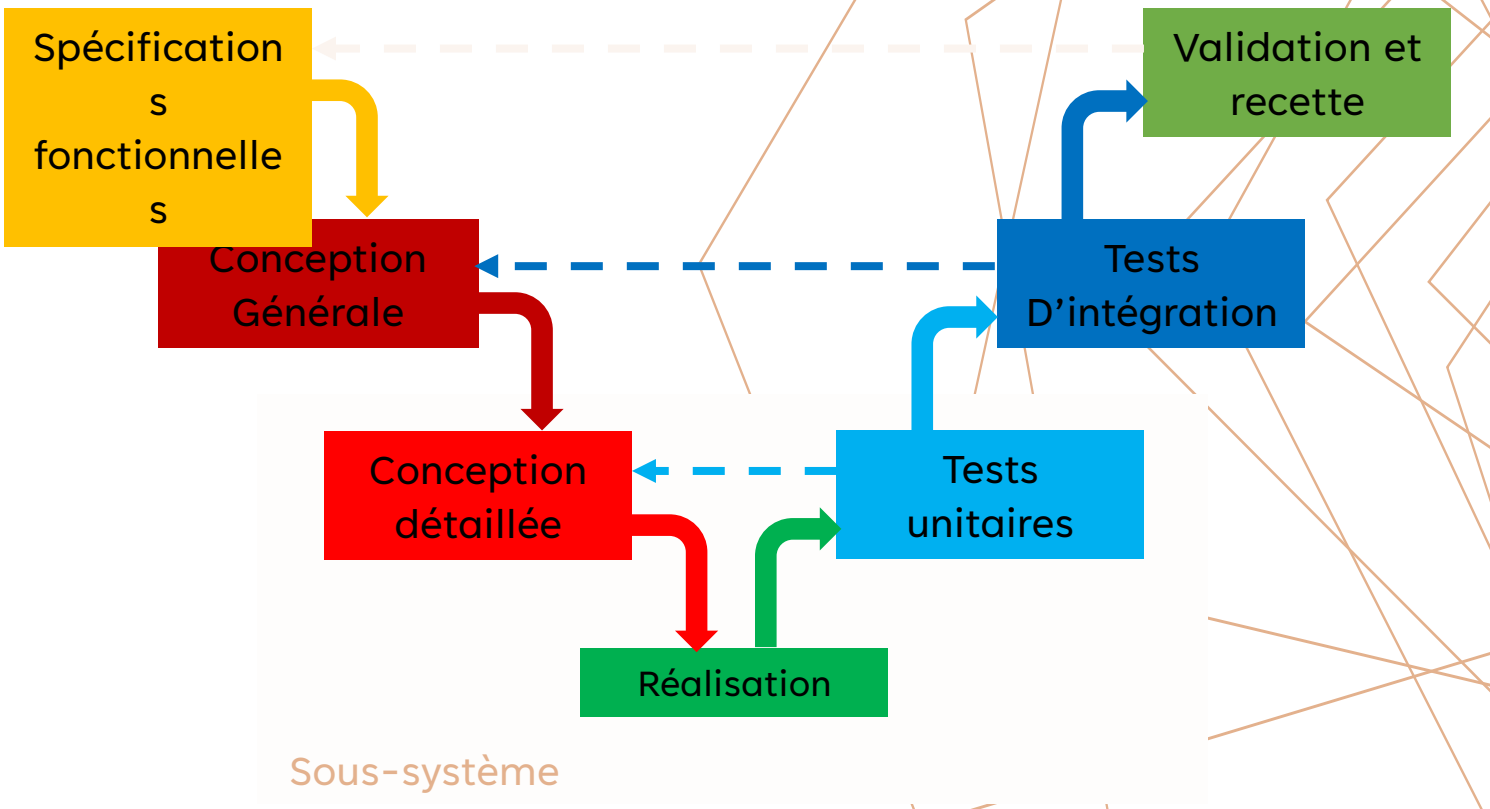
- Limite :

```
Lire (i,E, a[1],a[2])
Trouve<-faux
TantQue (i<=2) faire
  si a[i] =E alors
        Trouve <- vrai
  Sinon
        Trouve<- faux
  FinSI
  i<-i+1
FinTQ
```

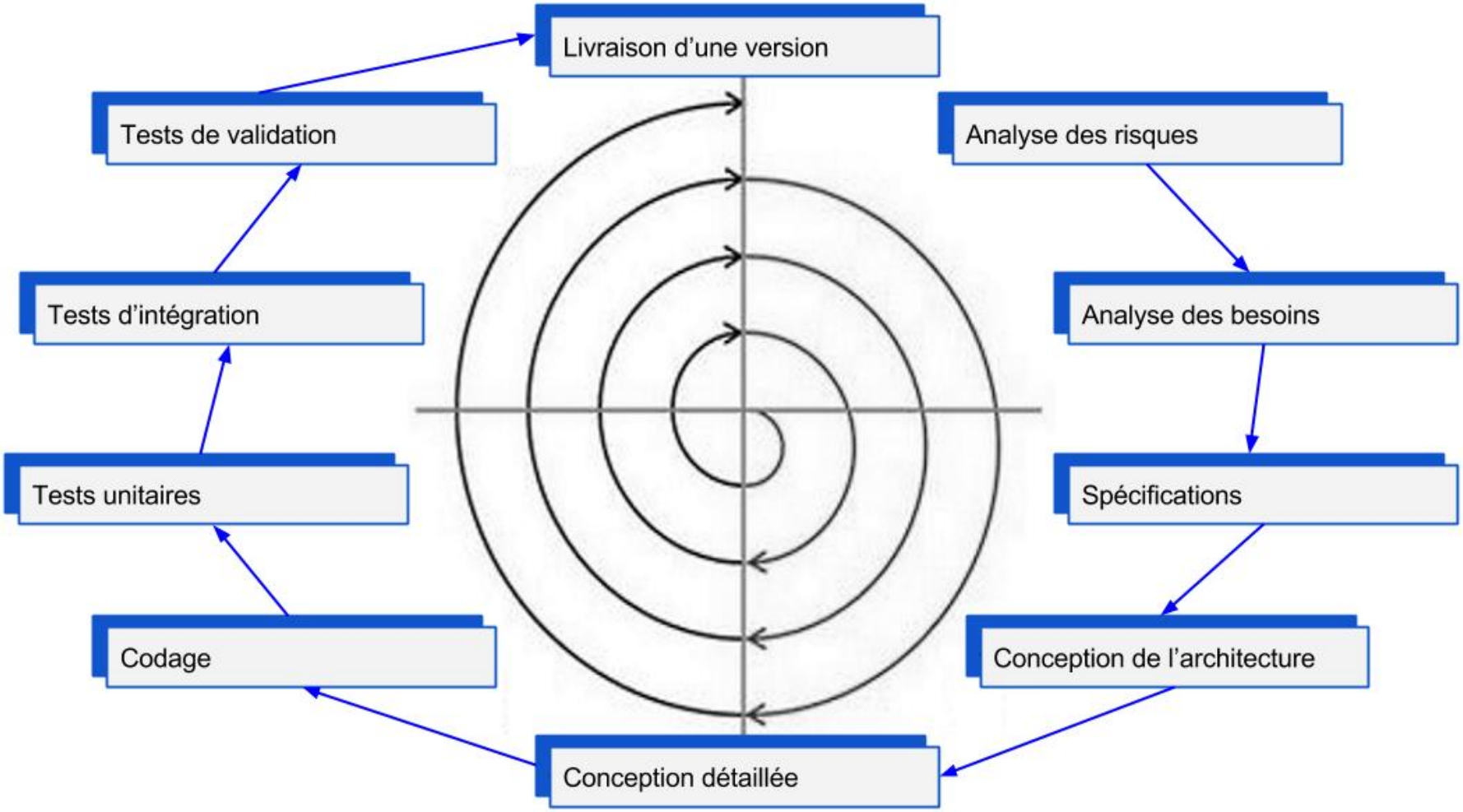
LE TEST DANS LE PROCESSUS DE DÉVELOPPEMENT



LE TEST DANS LE PROCESSUS DE DÉVELOPPEMENT



LE TEST DANS LE PROCESSUS DE DÉVELOPPEMENT



MERCI