Cours Qualité et Tests
Suite - Chapitre 3 : Tests

INSTITUT SUPERIEUR INFORMATIQUE ISI المعهد العالي للاعلامية

Responsable du cours :
Héla Hachicha

Année Universitaire : 2016 - 2017

2. Méthodes de test fonctionnel

- Le test fonctionnel vise à examiner le comportement fonctionnel du logiciel et sa conformité avec la spécification du logiciel

  ⇒ Sélection des Données de Tests (DT)

- Suppose que la structure interne du système n'est pas connue.
- Basé sur la spécification de l'interface du système et de ses fonctionnalités, il s'assure de leur respect par l'implémentation du système.
- Approprié pour le test du système mais également pour le test unitaire indépendant.

2. Méthodes de test fonctionnel

- Méthodes du test fonctionnel :

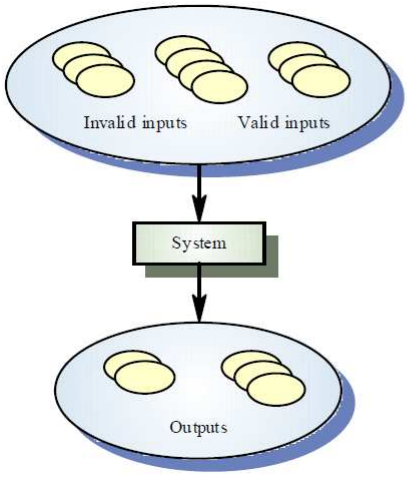
  2.1. Analyse partitionnelle des domaines des données d'entrée et test aux limites ➔ test déterministe

  2.2. Test combinatoire –Algorithmes Pairwise

  2.3. Test aléatoire

  2.4. Génération automatique de tests à partir d'une spécification

2.1. Analyse partitionnelle des domaines des données d'entrée et test aux limites



Une *classe d'équivalence* correspond à un ensemble de données de tests supposées tester le même comportement, c'est-à-dire activer le même défaut.

5

2.1. Analyse partitionnelle des domaines des données d'entrée et test aux limites

- Idée : diviser le domaine des entrées en un nombre fini de classes tel que le programme réagit pareil pour toutes les valeurs de la classe, donc il faut tester qu'une valeur par classe !

- Stratégie de test :
  1. Identifier les classes d'équivalence des entrées :
     - Sur la base des conditions sur les entrées/sortes.
     - En prenant des classes d'entrées valides et invalides.
  2. Définir un ou quelques DT pour chaque classe.

6

Règles de partitionnement des domaines

- Si la valeur appartient à un intervalle, construire :
  - une classe pour les valeurs inférieures,
  - une classe pour les valeurs supérieures,
  - n classes valides.

- Si la donnée est un ensemble de valeurs, construire :
  - une classe avec l'ensemble vide,
  - une classe avec trop de valeurs,
  - n classes valides.

- Si la donnée est une obligation ou une contrainte (forme, sens, syntaxe), construire :
  - une classe avec la contrainte respectée,
  - une classe avec la contrainte non-respectée

- .....

7

Exemple 1 : Valeur absolue

- Tester une fonction qui calcule la valeur absolue d'un entier.

Classes d'équivalence pour les entrées :

Condition	Classe valide	Classe invalide
nb. entrées	1	0, > 1
type entrée	int	string
valeurs valides	< 0, $>= 0$	

Données de test :
-10, 100, "XYZ", rien, 10 20

8

Exemple 2 : Calcul somme max

- Tester une fonction qui calcule la somme des premiers valeur entiers tant que cette somme reste plus petite que maxint. Sinon, une erreur est affichée. Si valeur est négatif, la valeur absolue est considérée.

Classes d'équivalence pour les entrées :

Condition	Classe valide	Classe invalide
nb. entrées	2	< 2, > 2
type entrée	int int	string int, int string
valeurs valides `value`	< 0, ≥ 0	
valeurs valides `maxint`	$>$ somme , $\leq$ somme	

Données de test :

maxint	value	return	maxint	value	return
100	10	55	10		error
100	-10	55	10 20	30	error
10	10	error	"XYZ"	10	error
10	-10	error	100	9.5	error

Message

Message

Analyse des valeurs limite

- Idée : les erreurs se nichent dans les cas limite, donc tester les valeurs aux limites des domaines ou des classes d'équivalence.

- Stratégie de test :
  - Tester les bornes des classes d'équivalence.
  - Tester les bornes du domaine des entrées.
  - Tester les entrées qui produisent les valeurs aux bornes pour les sorties.

Test aux limites

- Principe : on s'intéresse aux bornes des intervalles partitionnant les domaines des variables d'entrées :
  - pour chaque intervalle, on garde les 2 valeurs correspondant aux2 limites, et les 4 valeurs correspondant aux valeurs des limites ±le plus petit delta possible

    n ∈ 3 .. 15 ⇒v1 = 3, v2 = 15, v3 = 2, v4 = 4, v5 = 14, v6 = 16

  - si la variable appartient à un ensemble ordonnés de valeurs, on choisit le premier, le second, l'avant dernier et le dernier

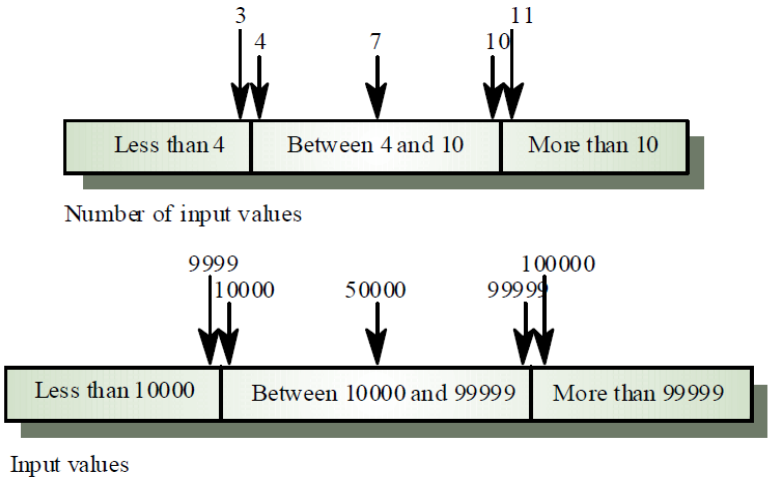
    n ∈ {-7, 2, 3, 157, 200} ⇒v1 = -7, v2 = 2, v3 = 157, v4 = 200

  - si une condition d'entrée spécifie un nombre de valeurs, définir les cas de test à partir du nombre minimum et maximum de valeurs, et des test pour des nombres de valeurs hors limites invalides.

Un fichier d'entrée contient 1-255 records, produire un cas de test pour 0, 1, 255 et 256.

11

Analyse des valeurs aux limites



12

Test aux limites –Types de données

- les données d'entrée ne sont pas seulement des valeurs numériques : caractères, booléens, images, son, …
- Ces catégories peuvent, en général, se prêter à une analyse partitionnelle et à l'examen des conditions aux limites :
  - True/ False
  - Fichier plein / Fichier vide
  - Trame pleine / Trame vide
  - Nuances de couleur
  - Plus grand / plus petit
  - ….

Test fonctionnel : Partition en classes d'équivalence

- **Exercice 1 :** Fonction qui attend un numéro de département entre 1 et 95 :

Validité des entrées	Classes d'équivalence	Représentant
entrées valides	[1, 95]	33
entrées invalides	< 1	-12
entrées invalides	> 95	401

Analyse aux bornes (Test aux limites)

Validité	Classes d'équiv.	Représentants	+large couverture
entrées valides	[1, 95]	1, 48, 95	1, 2, 48, 94, 95
entrées invalides	< 1	-12 ,0	-12, -1, 0
entrées invalides	> 95	96, 401	96, 97, 401

Test fonctionnel : Partition en classes d'équivalence

- **Exercice 2 :** Fonction qui attend une réponse oui/non :

Validité des entrées	Classes d'équivalence	Représentant
entrées valides	{oui}	"oui"
entrées valides	{non}	"non"
entrées invalides	autre (ni oui ni non)	"casimir"

Analyse aux bornes (Test aux limites)

Validité	Classes d'équivalence	Représentants
entrées valides	{oui}	"oui"
entrées valides	{non}	"non"
entrées invalides	autre (ni oui ni non)	"ouii", "mon", "oui ", "casimir"

Partitionnement de domaines : exemple

- Soit le programme suivant :

Un programme lit trois nombres réels qui correspondent à la longueur des cotés d'un triangle. Si ces trois nombres ne correspondent pas à un triangle, imprimer le message approprié. Dans le cas d'un triangle, le programme examine s'il s'agit d 'un triangle **isocèle**, **équilatéral** ou **scalène** et si son plus grand angle est **aigu**, **droit** ou **obtus** (< 90°, = 90°, > 90 °) et renvoie la réponse correspondante.

Classes d 'équivalence et Données de Test

	Aigu	Obtus	Droit
Scalène	6,5,3	5,6,10	3,4,5
Isocèle	6,1,6	7,4,4	√2,2,√2
Equilatéral	4,4,4	impossible	impossible

+ non triangle - 1,2,8

Test aux limites -Exemple

- Calcul des limites sur l'exemple du programme de classification des triangles

1, 1, 2	Non triangle
0, 0, 0	Un seul point
4, 0, 3	Une des longueurs est nulle
1, 2, 3.00001	Presque un triangle sans en être un
0.001, 0.001, 0.001	Très petit triangle
99999, 99999, 99999	Très grand triangle
3.00001, 3, 3	Presque équilatéral
2.99999, 3, 4	Presque isocèle
3, 4, 5.00001	Presque droit
3, 4, 5, 6	Quatre données
3	Une seule donnée
	Entrée vide
-3, -3, 5	Entrée négative

17

Méthodes de test fonctionnel -Exercice 3

- Supposons que nous élaborions un compilateur pour le langage BASIC. Un extrait des spécifications précise :

«*L'instruction FOR n'accepte qu'un seul paramètre en tant que variable auxiliaire. Son nom ne doit pas dépasser deux caractères non blancs; Après le signe = est précisée aussi une borne supérieure et une borne inférieure. Les bornes sont des entiers positifs et on place entre eux le mot-clé TO.* »

- Déterminer par analyse partitionnelle des domaines des données d'entrée les cas de test à produire pour l'instruction *FOR*.

18

Méthodes de test fonctionnel -Correction Exercice 1

DT obtenues par analyse partitionnelle pour l 'instruction *FOR* :

– FOR A=1 TO 10	cas nominal
– FOR A=10 TO 10	égalité des bornes
– FOR AA=2 TO 7	deux caractères pour la variable
– FOR A, B=1 TO 8	Erreur - deux variables
– FOR ABC=1 TO 10	Erreur - trois caractères pour la variable
– FOR I=10 TO 5	Erreur - Borne sup < Borne inf
– FOR =1 TO 5	Erreur - variable manquante
– FOR I=0.5 TO 2	Erreur - Borne inf décimale
– FOR I=1 TO 10.5	Erreur - Borne sup décimale
– FOR I=7 10	Erreur - TO manquant

19

Analyse partitionnelle et test aux limites – Synthèse

- L'analyse partitionnelle est une méthode qui vise à diminuer le nombre de cas de tests par calcul de classes d'équivalence
  - importance du choix de classes d'équivalence : risque de ne pas révéler un défaut
- Le choix de conditions d'entrée aux limites est une heuristique solide de choix de données d'entrée au sein des classes d'équivalence
  - cette heuristique n'est utilisable qu'en présence de relation d'ordre sur la donnée d'entrée considérée.
- Le test aux limites produit à la fois des cas de test nominaux (dans l'intervalle) et de robustesse (hors intervalle)

20

Analyse partitionnelle et test aux limites – **Combinaison des valeurs d'entrée**

1ère idée :

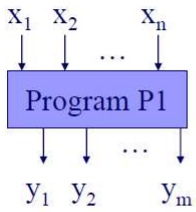
- On effectue le produit cartésien des domaines des entrées du programme.
- On utilise ce résultat comme ensemble de toutes les DTs et on détermine les sorties attendues pour chacune de ces DTs.

Défaut ➔ explosion combinatoire
➔ précis mais coûteux et long (parfois trop)

- Exemple : l'addition de 2 entiers de 32 bits génère 264 DTs !

21

Analyse partitionnelle et test aux limites – **Combinaison des valeurs d'entrée**



- Pb de la combinaison des valeurs des données d'entrée

Données de test pour la variable Xi :
$DTx_i = \{di_1, \ldots, di_n\}$

- Produit cartésien :
  $DTX_1 \times DTx_2 \times \ldots \times DTx_n$

=> Explosion du nombre de cas de tests

- Choix de classes d'équivalence portant sur l'ensemble des données d'entrée

22

Analyse partitionnelle et test aux limites – Combinaison des valeurs d'entrée

2ème idée :

- *On partitionne le produit cartésien en classes d'équivalence.*
- *Chaque classe d'équivalence correspond aux valeurs qui font l'objet d'un même traitement dans la spécification.*

1 classe d'équivalence ➔ 1 comportement du système

- *Exemples de comportements dans une spécification d'un ascenseur :*
  - *changer d'étage (monter ou descendre)*
  - *rester au même étage*
  - *éventuellement aller au rez-de-chaussée ou au dernier étage (test aux limites)*

Test aux limites -Evaluation

- Méthode de test fonctionnel très productive :
  - le comportement du programme aux valeurs limites est souvent pas ou insuffisamment examiné
- Couvre l'ensemble des phases de test (unitaires, d'intégration, de conformité et de régression)
- Inconvénient : caractère parfois intuitif ou subjectif de la notion de limite

⇒Difficulté pour caractériser la couverture de test.