Introduction aux tests du logiciel

F.X. Fornari
xavier.fornari@esterel-technologies.com
P. Manoury
pascal.manoury@pps.univ-paris-diderot.fr

2014

Test Fonctionnels

Aussi appelés test “boite-noire”

But

- vérification du comportement du logiciel par rapport aux spécifications (fonctions non conformes, manquantes, erreurs, ...)
- respect des contraintes (performances, memoire, delai, ...), et de qualité (portabilité, maintainabilité, documentation, ...)

Critère d'arrêt

Le nombre de tests nécessaires ne peut être connu a priori. Le **seul** élément certain est la spécification. Le testeur doit y trouver:

- les fonctions à implémenter

Critère d'arrêt

Le nombre de tests nécessaires ne peut être connu a priori. Le **seul** élément certain est la spécification. Le testeur doit y trouver:

- les fonctions à implémenter
- L'interface du logiciel

Critère d'arrêt

Le nombre de tests nécessaires ne peut être connu a priori. Le **seul** élément certain est la spécification. Le testeur doit y trouver:

- les fonctions à implémenter
- L'interface du logiciel
- Les contraintes:
  - delai, taille
  - hardware
  - sûreté
  - ...

Critère d'arrêt

Le nombre de tests nécessaires ne peut être connu a priori. Le **seul** élément certain est la spécification. Le testeur doit y trouver:

- les fonctions à implémenter
- L'interface du logiciel
- Les contraintes:
  - delai, taille
  - hardware
  - sûreté
  - ...

Les critères d'arrêt se font sur des mesures qualitatives.

Catégories de tests fonctionnels

Que test-t-on ? Comment ?

- Quoi ? couverture des tests
- Comment ? : analyse partitionnelle pour le test des fonctions de la spécification

Catégories de tests fonctionnels

- tests nominaux et aux limites: reliés aux fonctions du logiciel
  - Nominal: test la conformité par rapport à la spécification pour un comportement attendu
  - Limites: test du logiciel et de ses limites *définies*

Catégories de tests fonctionnels

- tests nominaux et aux limites: reliés aux fonctions du logiciel
  - Nominal: test la conformité par rapport à la spécification pour un comportement attendu
  - Limites: test du logiciel et de ses limites *définies*
- Robustesse: facteur de qualité
  - Liée à l'environnement: tests hors limites, charge/stress, défaut d'équipement externe, ...

Catégories de tests fonctionnels

- tests nominaux et aux limites: reliés aux fonctions du logiciel
  - Nominal: test la conformité par rapport à la spécification pour un comportement attendu
  - Limites: test du logiciel et de ses limites *définies*
- Robustesse: facteur de qualité
  - Liée à l'environnement: tests hors limites, charge/stress, défaut d'équipement externe, ...
- Tests de conformité: autres facteurs de qualités et/ou contraintes
  - perf, ergonomie, portabilité, ...

Analyse Partionnelle

Problème

Déterminer les jeux d'entrées

Analyse Partionnelle

Problème

Déterminer les jeux d'entrées

Première solution

Force brutale: produit cartésion des domaines d'entrées.
Défaut: potentiellement astronomique (addition $\rightarrow 2^{32}$).
Il faudrait des valeurs particulières représentatives.

Analyse Partionnelle

Problème

Déterminer les jeux d'entrées

Première solution

Force brutale: produit cartésion des domaines d'entrées.
Défaut: potentiellement astronomique (addition $\rightarrow 2^{32}$).
Il faudrait des valeurs particulières représentatives.

Seconde solution

Trouver des classes d'équivalences d'entrées: ensemble des entrées aboutissant au même comportement fonctionnel.
Par exemple: 2 tests pour l'addition: avec et sans débordement.

Analyse Partionnelle

Pour chaque fonction de la spécification:

- déterminer les entrées et leurs domaines

Analyse Partionnelle

Pour chaque fonction de la spécification:

- déterminer les entrées et leurs domaines
- en fonction de la partie contrôle de la spécification, découper chaque domaine en classes d'équivalence

Analyse Partionnelle

Pour chaque fonction de la spécification:

- déterminer les entrées et leurs domaines
- en fonction de la partie contrôle de la spécification, découper chaque domaine en classes d'équivalence
- Pour chaque classe:
  - sélectionner un représentant
  - déterminer le résultat attendu

Valeurs de sortie ou propriétés

Problème de l'oracle

- algorithme trop complexe (régulation en automatisme)
- toutes les entrées nécessaires par forcément accessibles au testeur (horloge système, positionnée par l'environnement, ...)

Partition

Partition

Soit un domain D, les ensembles C_1, ... C_n sont une partition en classes d'équivalence pour D si:

$$\begin{aligned} \forall i, j, C_i \cap C_j &= \emptyset \quad \text{règle 1: exclusion} \\ \bigcup_{i=1}^{n} C_i &= D \quad \text{règle 2: overlay} \end{aligned}$$

Règle 1 non satisfaite : La spécification n'est pas déterministe
Règle 2 non satisfaite : La spécification incomplète

Exemple

Programme calculant : $\sqrt{\frac{1}{x}}$

3 classes d'équivalence:

- $x < 0$
- $x = 0$
- $x > 0$

Une seule classe est valide

Détermination des classes d'équivalence

Différentes méthodes sont possibles:

- Langages formels $\Rightarrow$ détermination des chemins de la spécification

Détermination des classes d'équivalence

Différentes méthodes sont possibles:

- Langages formels $\Rightarrow$ détermination des chemins de la spécification
- Automates, réseaux de Petri $\Rightarrow$ parcours des automates, ...

Détermination des classes d'équivalence

Différentes méthodes sont possibles:

- Langages formels $\Rightarrow$ détermination des chemins de la spécification
- Automates, réseaux de Petri $\Rightarrow$ parcours des automates, ...
- Matrices cause/effet $\Rightarrow$ parcours des matrices

Détermination des classes d'équivalence

Différentes méthodes sont possibles:

- Langages formels $\Rightarrow$ détermination des chemins de la spécification
- Automates, réseaux de Petri $\Rightarrow$ parcours des automates, ...
- Matrices cause/effet $\Rightarrow$ parcours des matrices
- Langage naturel $\Rightarrow$ spécification re-modélisée en langage formalisé ou automate.

Spécification informelle

Si la spécification ne peut pas être testée, elle peut être rejetée, ou alors:

- un modèle doit être développé

Spécification informelle

Si la spécification ne peut pas être testée, elle peut être rejetée, ou alors:

- un modèle doit être développé
- ce modèle doit être validé par l'équipe de développement

Spécification informelle

Si la spécification ne peut pas être testée, elle peut être rejetée, ou alors:

- un modèle doit être développé
- ce modèle doit être validé par l'équipe de développement
- des classes d'équivalence peuvent être définies

Spécification informelle

Si la spécification ne peut pas être testée, elle peut être rejetée, ou alors:

- un modèle doit être développé
- ce modèle doit être validé par l'équipe de développement
- des classes d'équivalence peuvent être définies

Ce process de remodélisation permet souvent de détecter des problèmes très tôt:

- incohérences entre des parties de la spécification
- incomplétude des cas traités.

Test Nominaux

- sélection d'*une* valeur "intelligente" dans la classe d'équivalence
- varier les valeurs à l'intérieur d'un même intervalle (entropie)

Exemple de partition

Function: AbsoluteValProd
Inputs: E1, E2, integers
Outputs: S
Role: calcule la valeur absolue du produit des entrées

Classes d'équivalence pour les entrées

E1	E2
[Min_Int, -1]	[Min_Int, -1]
[0, Max_Int]	[0, Max_int]

Tests nominaux: choix "intelligent"

E1		E1	
[Min_Int,-1]	-734	[Min_Int,-1]	-525
[Min_Int,-1]	-7445	[0, Max_Int]	3765
[0, Max_Int]	7643	[Min_Int,-1]	-765
[0, Max_Int]	9864	[0, Max_Int]	3783

Test aux et hors limites fonctionnelles

- Tests aux limites fonctionnelles: sélection de valeurs aux bornes de chaque classe d'équivalence fonctionnelles
- Tests hors limites fonctionnelles: sélection de valeurs hors bornes de chaque classe d'équivalence fonctionnelles

Tests aux limites et hors limites

Si les entrées E1 et E2 sont dans le domaine [-100, 100].

Limites fonctionnelles

E1		E1	
[-100,-1]	-100	[-100,-1]	-57
[-100,-1]	-1	[0, +100]	64
[0, +100]	0	[-100,-1]	-5
[0, +100]	100	[0, +100]	98
[-100,-1]	-59	[-100,-1]	-1
[0, +100]	48	[-100,-1]	-100
[-100,-1]	-63	[0, +100]	0
[0, +100]	75	[0, +100]	100

Hors limites

E1		E1	
[-100,-1]	-234	[-100,-1]	-42
[0, +100]	174	[0, +100]	39
[-100,-1]	-84	[Min Int, -1]	-115
[0, +100]	48	[0, +100]	120

Tests de robustesse

Vérifier le comportement du logiciel face à des événements non spécifiés ou dans des situations dégradées:

- Tests en charge
- Tests des pannes des équipements externes
- Données incorrectes (lié aux tests hors limites),
- Mémoire insuffisante,
- Options incorrectes, ...

Teste en charge

Vérifier le comportement du logiciel en cas de stress du logiciel tel que:

- avalanche d'alarmes
- saturation des réseaux
- saturation des requêtes
- saturation des resources
- ...

Tests de pannes des équipements externes

Simuler des pannes sur les équipement en interface avec le logiciel afin de vérifier son comportement:

- arrêt inopiné de l'équipement
- débranchement brutal de l'équipement (USB...)
- changement brusque de valeurs
- ...

Tests de pannes des équipements externes: connaissances requises

Ces tests nécessitent une bonne connaissance du hardware afin de spécifier les bons modèles de défaillance des équipements.
Par exemple pour un interrupteur:

- collage à 0 ou 1,
- bagottements (acquisition intermittente)
- parasitage à différentes fréquences

Le test des interfaces

Le but des tests des des interfaces est double:

- vérifier les interfaces logicielles entre les composants d'un sous-système logiciel
- vérifier les interfaces physiques entre le logiciel est la machine cible (carte par ex, mais aussi drivers)

Le test des interfaces

Le but des tests des des interfaces est double:

- vérifier les interfaces logicielles entre les composants d'un sous-système logiciel
- vérifier les interfaces physiques entre le logiciel est la machine cible (carte par ex, mais aussi drivers)

Composants logiciels :Il y a deux types de tests à faire

- vérifier l'échange des données
- vérifier l'activation des composants

Conclusion pour les tests fonctionnels

Ce ne sont que quelques exemples: tout dépend énormément du métier pour lequel le logiciel est développé.
Les tests fonctionnels font intervenir l'environnement, alors que les tests structurels se concentrent sur l'intérieur du logiciel.

L'exemple du triangle

Spécification

Un programme a trois entiers en entrée. Ces entiers sont interprétés comme les longueurs des côtés d'un triangle.
Le programme indique s'il s'agit d'un triangle ordinaire (scalène), isocèle ou équilatéral

Question

Produire une suite de tests pour ce programme. Combien de cas ?

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième
10. 3 cas pour le test 9 avec permutations

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième
10. 3 cas pour le test 9 avec permutations
11. cas où les 3 entrées sont nulles

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième
10. 3 cas pour le test 9 avec permutations
11. cas où les 3 entrées sont nulles
12. 3 cas avec une valeur non entière

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième
10. 3 cas pour le test 9 avec permutations
11. cas où les 3 entrées sont nulles
12. 3 cas avec une valeur non entière
13. cas avec un nombre incorrect d'entrées

L'exemple du triangle

cas de tests possibles – GM Myers "The Art of Software Testing"

1. cas normal ordinaire (1,2,3 et 2,5,10 ne le sont pas. Pourquoi ?)
2. cas normal équilatéral
3. cas isocèle normal (2,2,4 n'est pas valide)
4. cas isocèle avec 3 permutations (3,3,4; 3,4,3, 4,3,3)
5. avec la valeur 0
6. avec une valeur négative
7. la somme de 2 entrées égale à la 3ième
8. 3 cas pour le test 7 avec permutations
9. cas où la somme des 2 entrées est supérieure à la 3ième
10. 3 cas pour le test 9 avec permutations
11. cas où les 3 entrées sont nulles
12. 3 cas avec une valeur non entière
13. cas avec un nombre incorrect d'entrées
14. pour chaque test, avez vous prédit le résultat ?