

Test du logiciel

1ère partie : Généralités

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

1

Questions

- ☞ Comment savez-vous si votre programme fonctionne correctement, s'il est fiable ?
- ☞ Comment le testez-vous ?
- ☞ Combien y consacrez-vous : de temps ? de budget ? de % développement total ?
- ☞ Jusqu'à quand le testez-vous ?
- ☞ Combien vous faudrait-il de temps en plus pour atteindre le "zéro défaut" ?
- ☞ Qu'est-ce que "tester" ?
- ☞ Qu'est-ce qu'une "métrique" ? En avez-vous déjà utilisé ? sur le code ? sur la conception ? sur la spécification ?
- ☞ Qu'est-ce que la "couverture de test" ? L'avez-vous déjà évaluée ?
- ☞ Avez-vous déjà utilisé un outil de test ?

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

2

Définitions du test


- ☞ IEEE729 (Standard Glossary of Software Engineering Terminology) :
« Le test est un processus consistant à exécuter ou évaluer un système ou un composant, par des moyens automatiques ou manuels, pour vérifier qu'il répond à des besoins spécifiés, ou pour identifier des différences entre les résultats attendus et les résultats obtenus »
- ☞ G.J. Myers (The Art of Software testing, 1979) : « Tester, c'est exécuter un programme dans l'intention d'y trouver des anomalies »

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

3

Terminologie

- ☞ Erreur
 - ☞ Défaut
 - ☞ Anomalie
- 

- ☞ Le processus de test consiste à rechercher des anomalies dans un programme,
pas à en diagnostiquer les causes,
et encore moins les corriger !
- ☞ Le processus de test ne consiste pas à prouver la correction du programme
- «Testing can reveal the presence of errors but never their absence » - Edsger W. Dijkstra, Notes on structured programming, Academic Press, 1972

- ☞ Terminologie complète :

<http://www.software-tester.ch/PDF-Files/Glossaire%20Version%201.0%20Francais.pdf>

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

4

L'évolution du test

- ☞ **Années 50 : Tester = mettre au point**
- ☞ **Années 60 : Montrer que le programme marche**
- ☞ **Années 70 : Montrer que le programme ne marche pas**
- ☞ **Années 80 : Montrer la qualité, l'évaluer**
- ☞ **Années 90 :**
 - **Contrôler la qualité : maîtriser, gérer, prévenir**
 - **Adaptation à l'objet, au Web**
- ☞ **Années 2000 :**
 - **plus d'automatisation (générateurs de tests, exécution automatique des tests ...)**
 - **XUnits**
 - **Test Driven Development (TDD)**
 - ...

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

7

Pourquoi tester ?

- ☞ **Coût de correction des anomalies**
- ☞ **Atteindre des niveaux de confiance et de qualité satisfaisants**
 - **Fiabilité**
 - **Robustesse**
 - **Conformité aux spécifications****tout en limitant l'effort de test**
 - ⇒ **niveaux préétablis, raisonnables**
- ☞ **Se rappeler que le test ne peut pas garantir la correction !**
«Testing can reveal the presence of errors but never their absence » -
Edsger W. Dijkstra, Notes on structured programming, Academic Press, 1972

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

8

Test = activité du processus de développement

- ☞ **50 à 60% de l'effort de développement total**
 - élaboration du matériel de test
 - déroulement des tests
 - analyse des résultats, décisions
 - documentation
- ☞ **répartie entre plusieurs phases**
- ☞ **dépend de la phase**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

12

Difficulté du test

- ☞ **Nature des problèmes**
- ☞ **Complexité des logiciels**
- ☞ **Quantité d'informations**
- ☞ **Test = processus destructif**
- ☞ **Frustrations dues à : points de vue, communication, planning, ressources, documentation, besoins, modifications ...**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

15

En résumé

- ☞ Test = planification, spécification, conception, construction, exécution et maintenance du matériel de test
- ☞ On teste pour diverses raisons
- ☞ Un bon test mène à une bonne qualité
- ☞ Le test fournit un produit
- ☞ Le produit logiciel et le matériel de test sont inter-dépendants

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

17

Test du logiciel

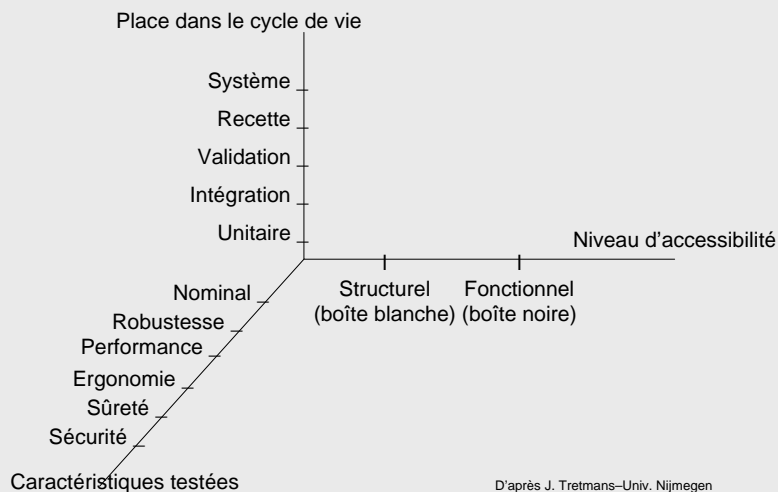
2ème partie : Techniques de test

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

18

Différents types de tests



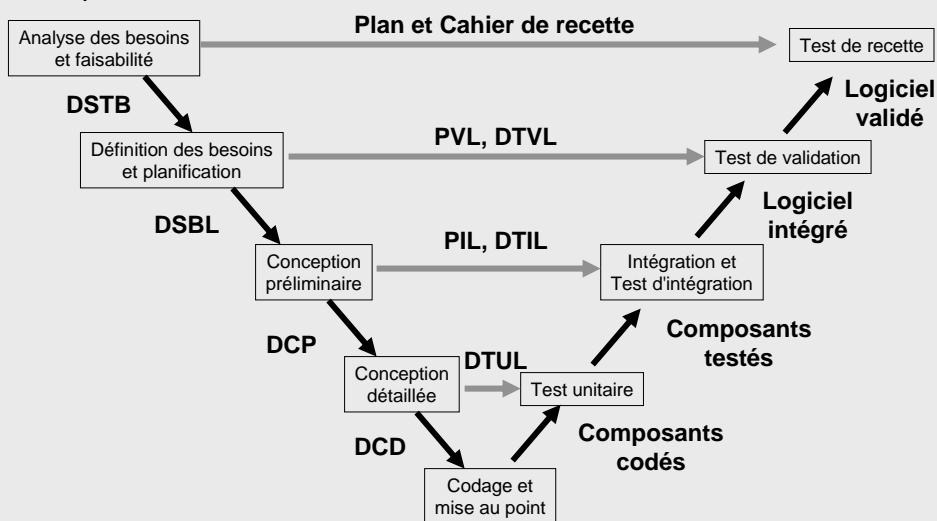
2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

19

Le test dans le cycle de vie

Cycle en V



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

21

Test et différents cycles de vie

☞ Cycle en V :

- Plans de test
 - » Objectifs du test
 - » Techniques de test
 - » Phases et planning

☞ Approches itératives :

- Pour chaque itération :
 - » Spécification / Conception / Codage / Test
- Plusieurs itérations :
 - » À chaque itération on teste tout \Rightarrow test de non régression sur les itérations précédentes

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

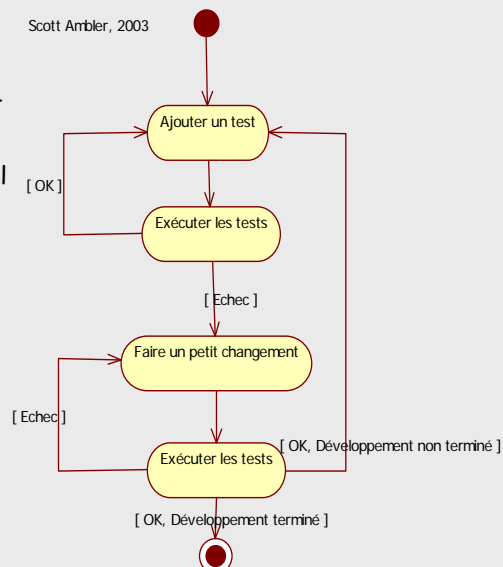
29

Test et différents cycles de vie

Scott Ambler, 2003

☞ Méthodes agiles :

- TFD : Test-First Development
 - » Définir les tests d'abord
 - » Développer le code minimal pour que les tests passent



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

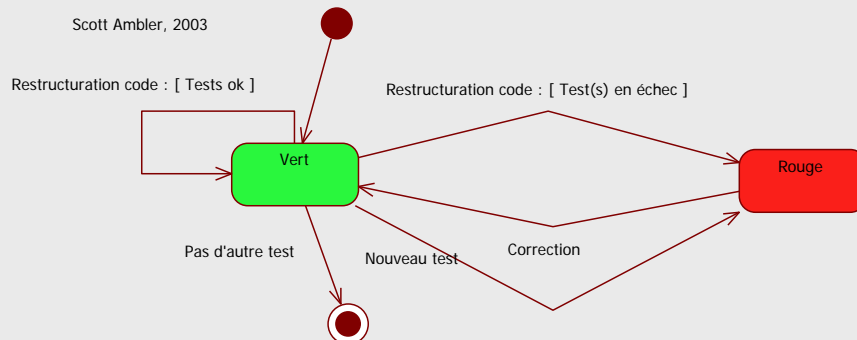
30

Test et différents cycles de vie

- TDD : Test Driven Development
 - » TDD = TFD + Restructuration (refactoring)

<http://www.agiledata.org/essays/tdd.html>
- Outils associés
 - » Xunit (Junit, Cunit, CppUnit ...)

Scott Ambler, 2003



2011-2012

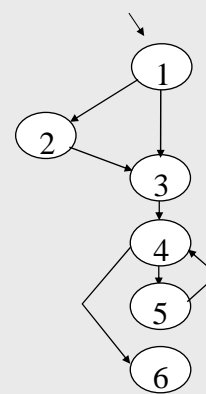
Henri Massié / L2 Informatique / UE Initiation Projet

31

Test structurel

Caractéristiques :

- dynamique
- boîte blanche
- fondé sur la structure du source du logiciel à tester :
 - » graphe de contrôle
 - » graphe d'appel
- méthode de sélection
 - » critères de couverture correspondants



Ex. de graphe de contrôle

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

37

Test structurel (suite)

☞ Critères de couverture

- du graphe de contrôle
 - » fondés sur le flot de contrôle :
 - ◆ toutes les instructions
 - ◆ tous les branchements
 - ◆ toutes les PLCS
 - ◆ toutes les combinaisons de 2, 3 ... PLCS
 - ◆ tous les chemins
 - ◆ ...
 - » fondés sur le flot de données
 - ◆ toutes les définitions de variables
 - ◆ tous les chemins définition-utilisation d'une variable
 - ◆ ...
- du graphe d'appel
 - ◆ tous les couples appelant/appelé
 - ◆ ...

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

38

Test structurel (suite & fin)

☞ Avantages

- en général facile à mettre en œuvre
- détecte facilement les erreurs commises
- arrêt sur objectif de couverture atteint
- très étudié, très outillé

☞ Inconvénients

- peut nécessiter le développement d'un matériel de test important pour soumettre le jeu de tests
- définition de l'oracle pas toujours facile
- **ne peut pas mettre en évidence des chemins manquants**
- non réutilisable

2011-2012

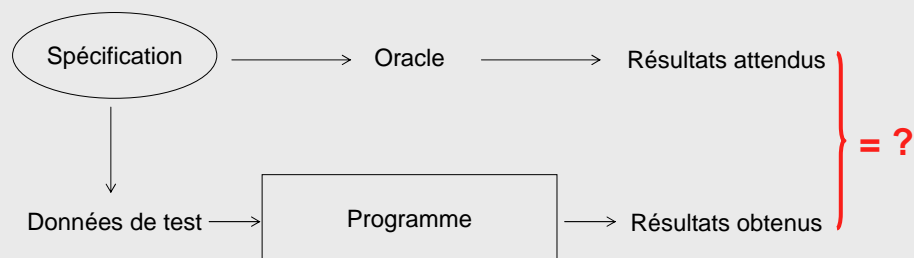
Henri Massié / L2 Informatique / UE Initiation Projet

39

Test fonctionnel

☞ Caractéristiques :

- dynamique
- boîte noire
- fondé sur la spécification du logiciel à tester
- méthode de sélection



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

40

Test fonctionnel

☞ Avantages

- **peut mettre en évidence des chemins manquants** (erreurs d'omission ou de spécification)
- réutilisable
- arrêt sur objectif de couverture atteint

☞ Inconvénients

- moins étudié, moins outillé

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

41

Complémentarité des tests

☞ Soit le code :

```
// somme de deux entiers  
public int somme(int x, int y) {  
    if (x==1000) {return x-y;}  
    else {return x+y;}  
}
```

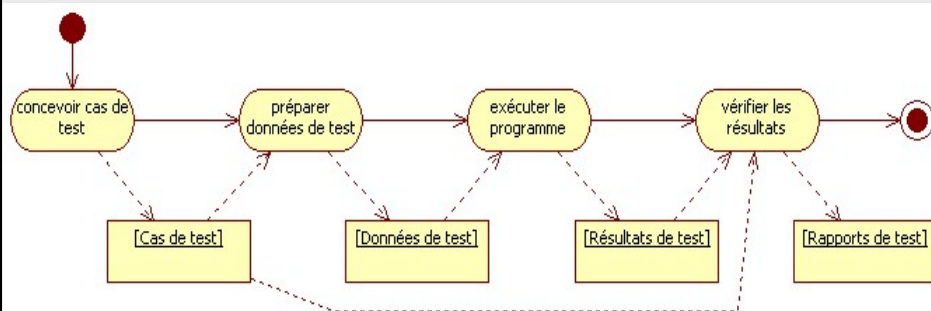
- ☞ Un test fonctionnel ou aléatoire détectera difficilement le défaut ; un test structurel le détectera très facilement
- ☞ Autre exemple : cf. chemins manquants

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

43

Le processus de test



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

53

Référentiel de test

(source : Legnard et al. - Industrialiser le test fonctionnel - novembre 2011)

- ☞ Cas de test et documentation associée
- ☞ [Scripts de test, pour exécution automatique]
- ☞ Données de test
- ☞ Résultats de test et documentation associée
- ☞ Anomalies détectées et leur état
- ☞ Liens de traçabilité vers le référentiel des exigences

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

56

Techniques de conception de tests

- ☞ Utilisées pour **construire le référentiel de test**
- ☞ Test structurel = **test « boîte blanche »**
 Test fonctionnel = test « boîte noire »

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

57

Test structurel

☞ **But : s'assurer que les éléments structurels importants sont exécutés**

☞ **3 phases :**

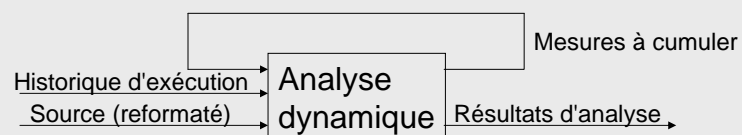
- déterminer les éléments structurels
ie décomposer statiquement le programme en "portions de code"
⇒ analyse statique
- exécuter le programme avec des jeux d'essais préalablement établis
- marquer les portions de code exécutées
afin de **mesurer la couverture de test**
⇒ analyse dynamique

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

58

Analyse dynamique : principe



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

72

Analyse dynamique : résultats

- ☞ **Analyse du flot de contrôle dynamique (déroulement du programme)**
 - Détection des erreurs de logique
 - Evaluation de la couverture de test (blocs, branchements, PLCS, etc.)
- ☞ **Profils d'exécution (par instruction)**
- ☞ **Analyse des jeux de données (utilité)**
- ☞ **Analyse dynamique du flot de données**
 - Détection des violations de domaines
- ☞ **Analyse détaillée des temps d'exécution**
 - en vue d'une optimisation (règle des 90/10)

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

73

Couvertures fondées sur le flot de contrôle

- ☞ **Critères :**
 - Couvrir, au moins une fois :
 - » Tous les blocs
 - » Tous les branchements (évite les masquages d'erreurs)
 - » Toutes les PLCS
 - » Tous les chemins indépendants (Mc Cabe)
 - » etc.

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

81

Couverture des blocs, branchements et PLCS

Source

N : Natural ;
Natural_IO.get(N) ;

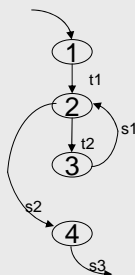
for l in 1..N loop

...

end loop ;

Text_IO.put("Fin") ;

Graphe de contrôle



PLCS

p1 : 1&2, 4
p2 : 1&2&3, 2
p3 : 2,4
p4 : 2&3,2
p5 : 4, -1

Jeux d'essai	Blocs exécutés	Branchements exécutés	PLCS exécutées	Remarques
N=3	1, 2, 3, 2, 3, 2, 3, 2, 4, -1	t1, t2, s1, t2, s1, t2, s1, s2, s3	p2, p4, p4, p3, p5	100% blocs 100% branchements 80% PLCS
N=0	1, 2, 4, -1	t1, s2, s3	p1, p5	Nécessaire pour 100% PLCS (en cumulé)
N=1	1, 2, 3, 2, 4, -1	t1, t2, s1, s2, s3	p2, p3, p5	Redondant avec N=3

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

82

Couvertures fondées sur le flot de contrôle

☞ TER1 = Nombre de blocs exécutés / Nombre total de blocs exécutables

☞ TER2 = Nombre de branchements exécutés / Nombre total de branchements

☞ TER3 = Nombre de plcs exécutées / Nombre total de plcs

[coursTest Couv flot controle.pdf](#)

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

83

Autres critères

- ☞ **Chemins limites et chemins intérieurs à une boucle (0 et 1 exécution de la boucle)**
- ☞ **Combinaisons de PLCS :**
 - TER 4 = nombre de chemins composés de 2 PLCS exécutées / nombre total de chemins composés de 2 PLCS
 - TER n+1 = Nombre de chemins composés de n-1 PLCS exécutées / nombre total de chemins composés de n-1 PLCS

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

84

Couvertures fondées sur le flot de données

- ☞ **Critères :**
 - Couverture de tous les chemins de p-utilisation (usage de la variable dans des prédicats d'instruction de décision)
 - Couverture de tous les chemins de c-utilisation (usage de la variable dans des calculs ou en indice de tableau)
 - Couverture de tous les chemins de p-utilisation et de quelques chemins de c-utilisation
 - Couverture de tous les chemins de c-utilisation et de quelques chemins de p-utilisation
 - Couverture de tous les chemins d'utilisation
 - Couverture de tous les chemins DU (définition/utilisation)
 - Couverture de toutes les définitions

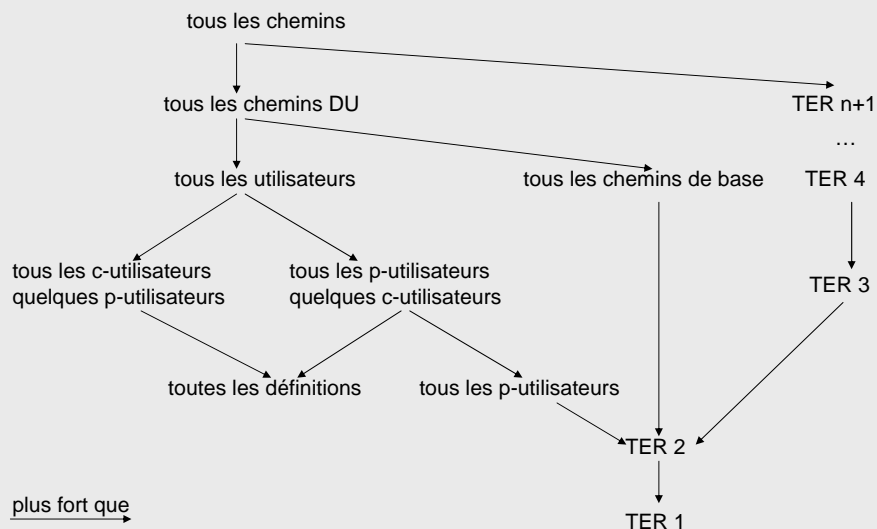
[coursTest Couv flot donnees.pdf](#)

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

85

Hiérarchie de critères de couverture



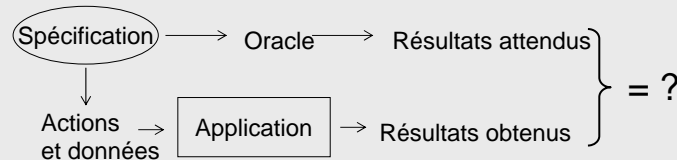
2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

86

Test fonctionnel = test « boîte noire »

- Aucune connaissance de la structure du code de l'application testée
- Elle est vue au travers de ses seules interfaces : **actions et contrôles** réalisables d'un point de vue externe :
 - elle est stimulée à travers ses **points de contrôle**
 - les résultats sont vérifiés à travers ses **points d'observation**



- L'infrastructure de test doit permettre de maîtriser l'environnement de l'application testée
- L'exécution automatique peut nécessiter l'écriture de code spécifique pour le test (~préambule/postambule, cf JUnit)

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

93

Test fonctionnel = test « boîte noire »

☞ Processus général de test « boîte noire » :

- **Analyse des besoins et des spécifications**
- **Choix d'entrées valides** sur la base de la spécification pour vérifier que l'application les traite correctement. Il faut aussi **choisir des entrées invalides** pour vérifier que l'application les détecte et les gère correctement
- Détermination des **résultats attendus** pour ces entrées
- **Construction des tests** avec ces entrées
- **Exécution des tests**
- **Comparaison des résultats** obtenus avec les résultats attendus
- **Détermination vis-à-vis du bon fonctionnement** de l'application

☞ Applicable à tout niveau du développement, mais :

Unitaire Intégration **Système Recette**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

94

Partitionnement en classes d'équivalence

- ☞ **But** : permettre de **produire un référentiel de test réduit**, en réduisant l'ensemble des "données" possibles à un nombre limité de combinaisons "intéressantes"
- ☞ **Subdivision des "données" en classes d'équivalence** : l'ensemble des valeurs d'une même classe déclenche le même comportement
 - Exemple : Connexion : 2 comportements différents suivant que utilisateur enregistré ou pas, donc 2 classes d'équivalence sur les utilisateurs
- ☞ Le choix des tests est guidé par :
 - le choix d'un **représentant par classe**
 - la couverture des comportements à tester
- ☞ La classification porte sur :
 - le **domaine des entités métier**
 - les **données en entrée des actions**
 - les **résultats attendus**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

97

Partitionnement en classes d'équivalence

☞ Démarche :

- Identifier les **classes d'équivalence**
- Identifier les **cas de test**
- Compléter les cas de test par la spécification des **résultats attendus** correspondants

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

98

Partitionnement en classes d'équivalence

☞ Identifier les classes d'équivalence

- A partir des **règles métier**
 - » Exemple : malus pour l'achat de véhicules neufs polluants

Taux d'émission de CO ₂ /km	Montant du malus en 2012
Entre 141 et 150 grammes	200 €
Entre 151 et 155 grammes	500 €
Entre 156 et 180 grammes	750 €
Entre 181 et 190 grammes	1300 €
Entre 191 et 230 grammes	2300 €
Au-delà de 230 grammes	3600 €

donc 6 classes d'équivalence

- **Tenir compte des données invalides**, qui mènent à un comportement d'erreur et que l'application doit traiter sans défaillance (un message d'erreur en général)
 - » Exemple : une valeur sous le seuil minimal, une au dessus du seuil maximal

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

99

Partitionnement en classes d'équivalence

- Portent sur **différents types de domaines** :
 - » **Numériques** : cas le plus simple
 - » **Énumérés** : si un sous-ensemble des éléments correspond à un cas qui sera traité de la même façon, ou si les conditions d'un état d'entrée d'action correspondent à un ensemble de valeurs, il faut identifier un représentant valide et un représentant invalide
 - » **Structurés** : il faut déterminer des caractéristiques communes, auxquelles on associe des comportements types ; chaque caractéristique constitue une classe
- En résumé, il faut :
 - » Identifier les **exigences** sur les "données", dans la spécification
 - » Traduire chaque exigence en un **domaine de valeurs** possibles
 - » Partager le domaine en **sous-domaines disjoints, recouvrant le domaine**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

100

Partitionnement en classes d'équivalence

Exigence de type	Classes valides	Classes invalides
intervalle de valeurs	<ul style="list-style-type: none"> une classe de valeurs valides 	<ul style="list-style-type: none"> une classe de valeurs inférieures, si représentables une classe de valeurs supérieures, si représentables
nombre limité de valeurs	<ul style="list-style-type: none"> une classe pour un nombre valide de valeurs 	<ul style="list-style-type: none"> une classe pas assez de valeurs une classe trop de valeurs
ensemble de valeurs à priori traitées différemment	<ul style="list-style-type: none"> une classe par valeur valide ... 	<ul style="list-style-type: none"> une classe pour toutes les valeurs invalides
obligation ou contrainte	<ul style="list-style-type: none"> une classe contrainte respectée 	<ul style="list-style-type: none"> une classe contrainte non respectée

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

101

Partitionnement en classes d'équivalence

☞ Identifier les cas de test, pour couvrir

- les classes **valides** non encore couvertes :
 - » Etablir un nouveau cas de test qui en recouvre le **plus possible**, jusqu'à ce que toutes soient couvertes
- les classes **invalides** non encore couvertes :
 - » Etablir un nouveau cas de test qui en recouvre **une (et une seule)**, jusqu'à ce que toutes soient couvertes

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

102

Exemple 1

☞ Problème :

- $F(x) = \sqrt{1/x}$

☞ 1 domaine :

- réel

☞ 2 contraintes :

- $x \neq 0$
- $1/x \geq 0$

☞ 2x2 classes disjointes :

- $x \neq 0$ (valide), $x = 0$ (invalid)
- $x \geq 0$ (valide), $x < 0$ (invalid)

☞ 3 cas de test :

- un réel positif (les 2 valides)
- 0
- un réel négatif

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

103

Exemple 2

☞ Détermination des caractéristiques d'un triangle

- le programme lit trois nombres réels représentant les longueurs des côtés d'un triangle
- il doit déterminer si ce triangle est scalène, isocèle ou équilatéral et afficher le résultat

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

104

Exemple 2 (suite)

☞ Classes d'équivalence

- 1 exigence nombre de valeurs
- Pour chaque valeur :
 - » 1 domaine : réel
 - » 1 contrainte intervalle : « représenter la longueur d'un côté »

Exigences	Classes valides	Classes invalides
nombre limité de valeurs	une classe 3 valeurs	une classe moins de 3 valeurs une classe plus de 3 valeurs
intervalle $0 < l_1 < l_2 + l_3$	une classe $0 < l_1 < l_2 + l_3$	une classe $l_1 \leq 0$ une classe $l_1 \geq l_2 + l_3$
de même pour l_2 et l_3

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

105

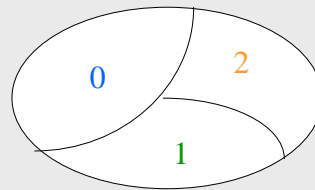
Exemple 2 (suite et fin)

☞ Cas de test

- 1 seul cas de test pour couvrir les 4 classes valides
» ex : (3, 4, 5)
- 8 cas de test pour couvrir les 8 classes invalides
» ex : (1, 2, 4) ; (-1, 2, 4) ...

☞ Sous-classes de triangles valides

- partitionnement suivant le résultat
- résultat fonction des égalités
(aucune, une ou deux)
⇒ 3 sous-classes
- 5 cas de test pour couvrir toutes les classes valides
» ex : (3, 4, 5) ; (3, 3, 4), (3, 4, 3), (4, 3, 3) ; (3, 3, 3)



2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

106

Partitionnement en classes d'équivalence

☞ Avantages

- couvre bien les domaines des entrées

☞ Inconvénients

- n'explore pas les combinaisons d'exigences sur les entrées
- ne permet pas de détecter certains défauts de codage courants : \leq à la place de $<$...

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

109

Approche « par paires »

- ☞ Tester un fragment des combinaisons de valeurs qui garantissent que chaque combinaison de deux valeurs est testée
- ☞ Exemple : 4 variables avec 3 valeurs possibles chacune
 - $3 \times 3 \times 3 \times 3 = 81$ combinaisons possibles
 - Toutes les paires : 9 combinaisons suffisent
- ☞ Avantages :
 - On teste des combinaisons
- ☞ Inconvénients :
 - La combinaison choisie n'est peut-être pas celle qui détectera le défaut

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

110

Test aux limites

- ☞ Extension du partitionnement par classes d'équivalence
 - ☞ Fondé sur l'étude des limites des domaines de définition des entrées, mais aussi des sorties
 - ☞ Conditions aux limites = situations
 - exactement sur,
 - juste avant
 - et juste après
- les frontières des classes d'équivalence correspondantes**

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

111

Test aux limites (suite)

☞ Identifier les cas de test

- Pour chaque classe d'équivalence, identifier les conditions aux limites
 - » Exemple : malus pour l'achat de véhicules neufs polluants
- Valeurs limites :
- 139, 140, 141 ; 149, 150, 151 ; 154, 155, 156 ; 179, 180, 181 ; 189, 190, 191 ;
229, 230, 231
- très grande valeur
- 0 ?
- Etablir un nouveau cas de test qui en recouvre **une (et une seule)**, jusqu'à ce que toutes soient couvertes

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

112

Test aux limites (suite)

Exigences de type	Cas de test valides	Cas de test invalides
intervalle de valeurs a..b	<ul style="list-style-type: none"> • a • b • succ(a) • pred(b) 	<ul style="list-style-type: none"> • pred(a) • succ(b)
nombre limité de valeurs	<ul style="list-style-type: none"> • minimum • maximum • minimum+1 • maximum-1 	<ul style="list-style-type: none"> • minimum-1 • maximum+1
ensemble ordonné de valeurs (tableau, liste, fichier ...)	<ul style="list-style-type: none"> • premier • dernier • second • avant-dernier 	<ul style="list-style-type: none"> • avant premier • après dernier

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

113

Exemple : triangle

(3, 2, 1)	non triangle : plat
(0, 0, 0)	non triangle : point
(0, 1, 1)	non triangle : une longueur nulle
(3.0001, 2, 1)	non triangle, mais presque
(0.0001, 0.0001, 0.0001)	très petit triangle, équilatéral
(99999, 99999, 99999)	très grand triangle, équilatéral
(3.0001, 3, 3)	presque équilatéral
(2.9999, 3, 4)	presque isocèle
...	

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

114

Test aux limites (suite et fin)

☞ Avantages

- l'une des méthodes fonctionnelles les plus efficaces
- couvre une large gamme d'erreurs
- utilisable dans toute phase de test
- utilisable en tests de charge, de performances, de précision

☞ Inconvénients

- difficulté de formaliser la notion de limite
- n'explore pas les combinaisons d'exigences sur les entrées

2011-2012

Henri Massié / L2 Informatique / UE Initiation Projet

115