Institut Supérieur des Etudes Technologiques de Radès

Master: Développement des applications mobiles

Test et performance des logiciels

CHAPTER 4:

TEST DESIGN TECHNIQUES

S. BEJI, Janvier 2020

Plan

4.1 The Test Development Process

4.2 Categories of Test Design Techniques

4.3 Specification-based or Black-box Techniques

4.4 Structure-based or White-box Techniques

4.5 Experience-based Techniques

4.6 Choosing Test Techniques

4.1 The Test Development Process

Background :

- A test condition is defined as an item or event that could be verified by one or more test cases (a function, transaction)

- A test case consists of a set of input values, execution preconditions, expected results and execution postconditions, defined to cover a certain test objective(s) or test condition(s).

4.1 The Test Development Process

- Expected results should be produced as part of a test case and include outputs, changes to data and states , and any other consequences of the test.

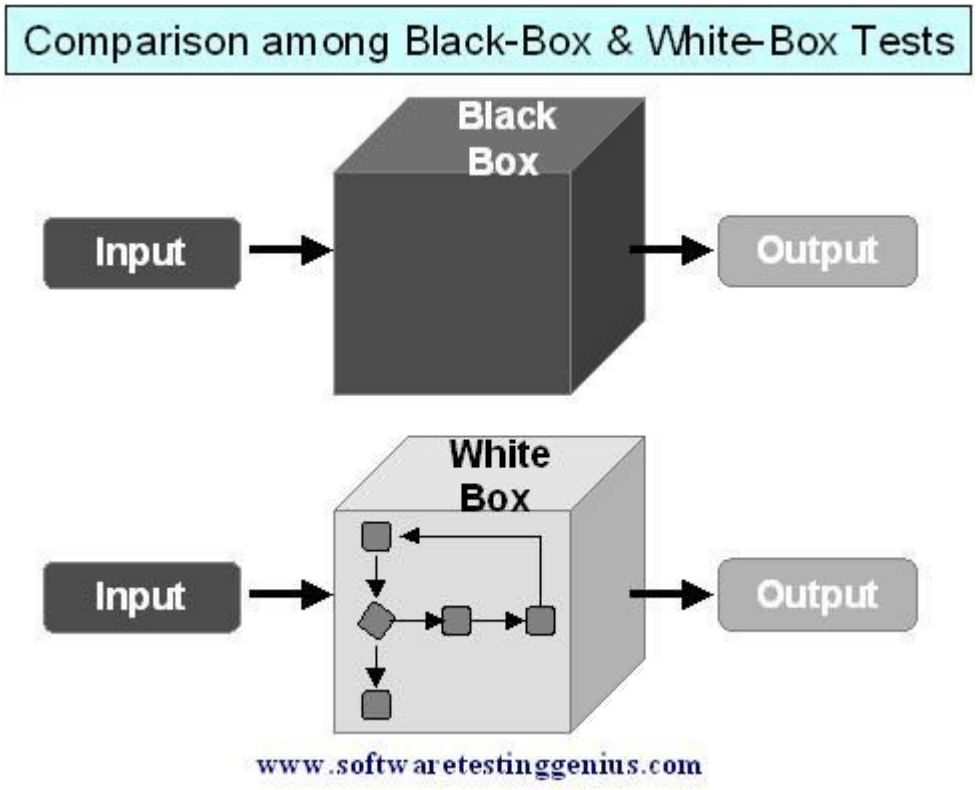
- During test implementation the test cases are developped, implemented, prioritized and organized in the test procedure specification (IEEE STD 829-1998). The test procedure specifies the sequence of actions for the execution of a test.

4.2 Categories of Test Design Techniques

Background :

- The purpose of a test design technique is to identify test conditions, test cases, and test data.
- Classic distinction to denote test techniques as :
  - Black-box.
  - White-box.
- Black-box testing, by definition, does not use any information regarding the internal structure of the component or system to be tested.
- White-box test design techniques (also called structural or structure-based techniques) are based on an analysis of the structure of the component or system.

4.2 Categories of Test Design Techniques



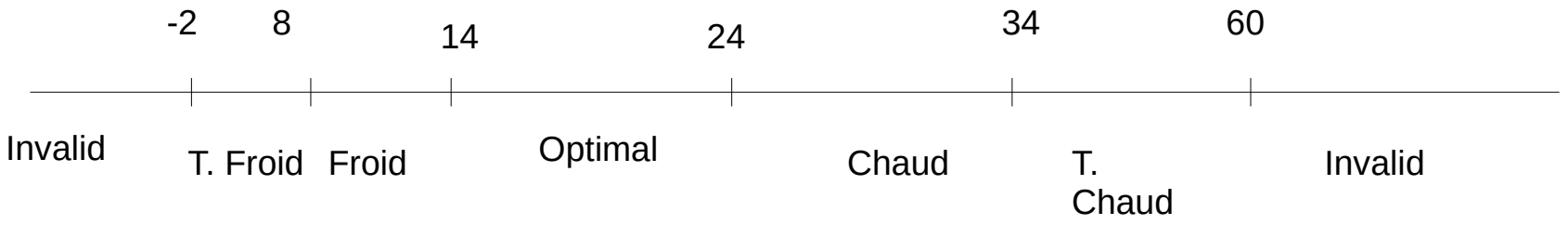
4.3 Specification-based or Black-box Techniques

- *4.3.1 Equivalence Partitioning* :

- In equivalence partitioning, inputs to the software or system are devided into groups that are expected to exhibit similar behavior, so they are likely to be processed in the same way.
- Tests can be designed to cover all valid and invalid partitions.
- Equivalence partitioning can be used to achieve input and output coverage goals.

4.3 Specification-based or Black-box Techniques

4.3.1 Equivalence Partitioning Example

- String : CheckTemperatureState (float temperature)
- 

4.3 Specification-based or Black-box Techniques

String : CheckTemperatureState (float temperature)

Invalid :]-oo , -2] / [60, +oo[

Valid : [8 ; 14] / [14, 24] / [24 , 34] / [34, 60]

4.3 Specification-based or Black-box Techniques

String : CheckTemperatureState (float temperature)

Invalid : -1.5 / 85.3

Valid : 11.3 / 22.8 / 26.2 / 47

4.3 Specification-based or Black-box Techniques

Moyennant la même technique, identifier les classes d’équivalence pour la fonction :

String **getMention** (Float moyenneEtudiant)

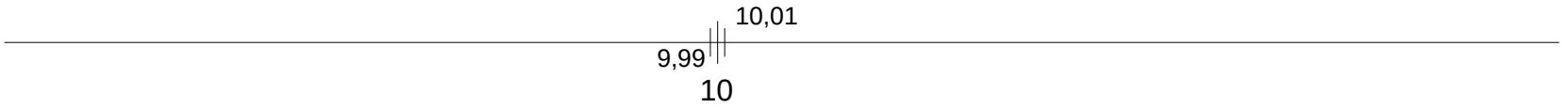
4.3 Specification-based or Black-box Techniques

- *4.3.2 Boundary Value Analysis*

- Behavior at the edge of each equivalence partition is more likely to be incorrect than behavior within the partition, so boundaries are an area where testing is likely to yield defects.
- The maximum and minimum values of a partition are its boundary values.

4.3 Specification-based or Black-box Techniques

- *4.3.2 Boundary Value Analysis Example*



4.3 Specification-based or Black-box Techniques

- *Idem pour la fonction getMention*

4.3 Specification-based or Black-box Techniques

- *4.3.3 Decision Table Testing :*
- Decision tables are a good way to capture system requirements that contain logical conditions, and to document internal system design. The may be used to record complex business rules that a system is to implement.
- The coverage standard commonly used with decision table testing is to have at least one test per column in the table.

4.3 Specification-based or Black-box Techniques

- *4.3.3 Decision Table Testing*

Condition	1	2
Condition 1	T	F
Condition 2	F	T
Action		
Action 1	Y	Y
Action 2	Y	N

4.3 Specification-based or Black-box Techniques

- *4.3.3 Decision Table Testing Example*

Condition	1	2	3	4	...	8
Temperature > 30	T	T	F	F		F
Humidité > 60 %	T	F	T	F		F
Transpiration	T	F	-	T		F
Action						
Activer climatisation	Y	Y	N	Y		N
Activer déshumidification	Y	N	Y	N		N

4.3 Specification-based or Black-box Techniques

Un commerçant veut booster ses ventes alors il décide de mettre en place les mécanismes suivants:

Pour toute vente supérieure à 300 DT, la livraison sera gratuite. Si le paiement est en ligne il y aura une remise de 3%.

Pour tous les achats entre 22H et ùinuit, une remise de 3% sera accordée également.

Si le client a un abonnement VIP, il lui donne la possibilité à une remise systématique de 2%.

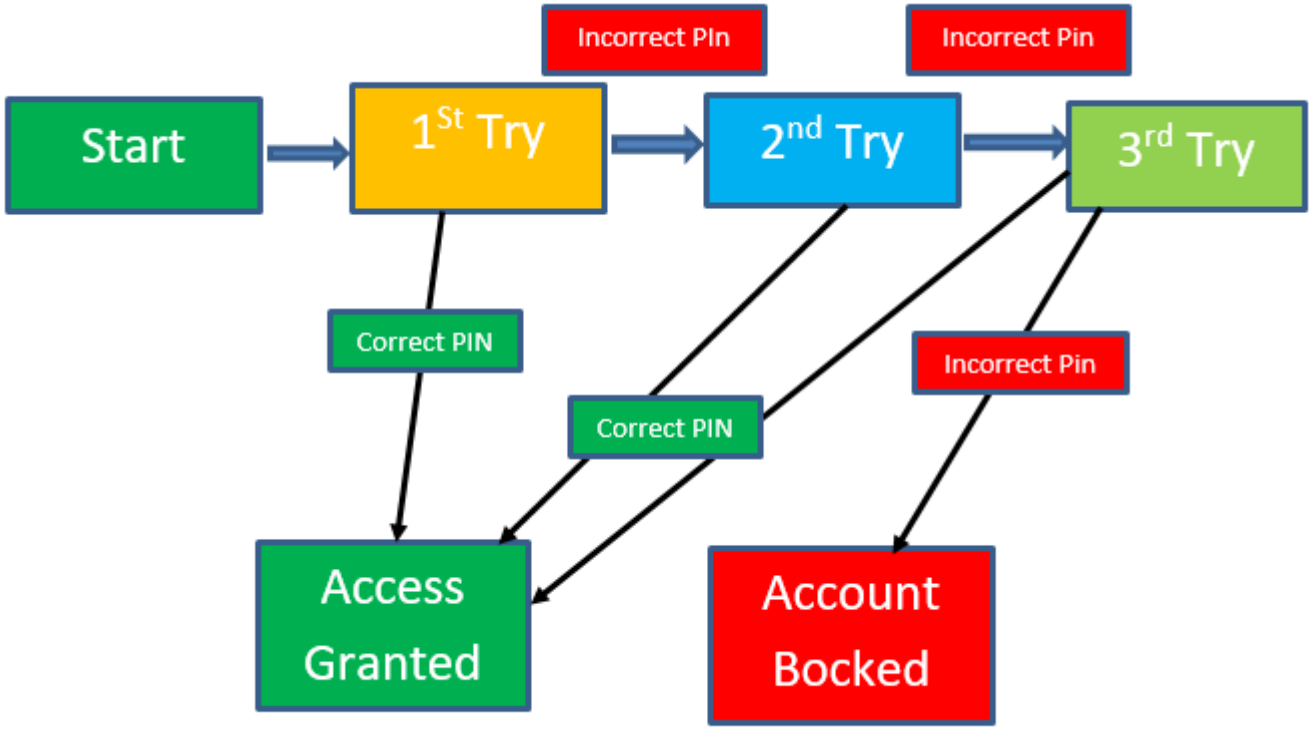
Dresser la table de décision.

4.3 Specification-based or Black-box Techniques

- *4.3.4 State Transition Testing :*
- It allows the tester to view the software in terms of its states, transitions between states, the inputs or events that trigger state changes (transition) and the actions which may result from those transitions.
- Tests can be designed to cover a typical sequence of states, to cover every state, to exercise every transition, to exercise specific sequences of transitions or to test invalid transitions.
- State transition testing is much used within the embedded software industry and technical automation in general.

4.3 Specification-based or Black-box Techniques

Banking Card Example



4.3 Specification-based or Black-box Techniques

Banking Card Example

State Transition Table

	Correct PIN	Incorrect PIN
S1) Start	S5	S2
S2) 1st attempt	S5	S3
S3) 2nd attempt	S5	S4
S4) 3rd attempt	S5	S6
S5) Access Granted	-	-
S6) Account blocked	-	-

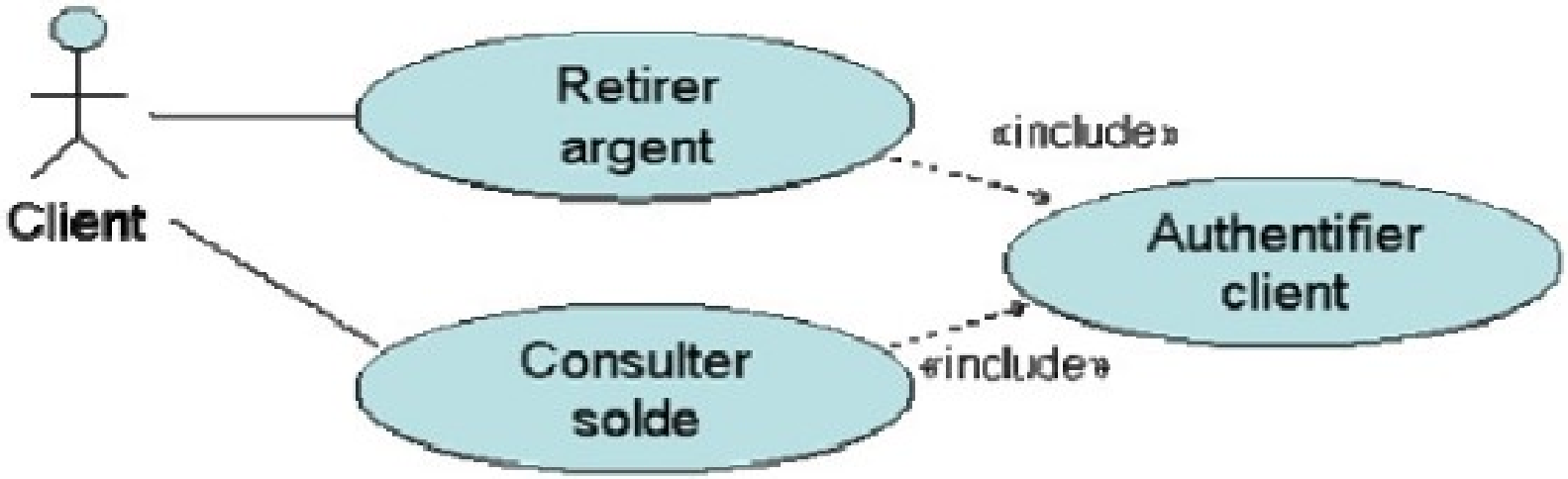
4.3 Specification-based or Black-box Techniques

- *4.3.5 Use case Testing:*

- Tests can be derived from use cases.
- Use cases are very useful for designing acceptance tests with customer/user participation.
- They also help uncover integration defects caused by the interaction and interference of different components, which individual component testing would not see.

4.3 Specification-based or Black-box Techniques

4.3.5 Use case Testing:



Quels seront les **Test Case** relatif au **Use Case Retirer argent** sachant que la banque peut tolérer 100DT de solde négatif, sans prélèvement d'intérêt.

4.3 Specification-based or Black-box Techniques

Test Case	Description	Préconditions	Etapes	Expected_Result	Post-condition
#1	Retirer argent avec solde suffisent	Solde=300	s'authentifier Mnt_retrait=100	Retrait de 200 DT	Solde=100
#2	Retirer argent avec solde insuffisent	Solde=300	s'authentifier Mnt_retrait=500	Retrait impossible, saisir un autre montant	Solde=300
#3	Retirer argent avec solde insuffisent mais > -100 DT	Solde=300	s'authentifier Mnt_retrait=350	Retrait de 350 DT	Solde=-50 Création Notification_Alimentation_Compte

4.4 Structure-based or white-box Techniques

Background :

Structure-based or white-box testing is based on an identified structure of the software or the system, as seen in the following examples :

1) *Component level* : the **structure of a software** component, i.e., statements, decisions, branches or even distinct paths.

2) *Integration level* : The structure may be a call tree (a **diagram in which modules call other** modules)

3) System level : The structure may be a menu structure, **business process** or web page structure.

4.4 Structure-based or White-box techniques

- *4.4.1 Statement Testing and coverage :*

- In component testing, statement coverage is the assessment of the **percentage of executable statements that have been exercised** by a test case suite.

4.4 Structure-based or White-box techniques

```
1   if (a || b)
2      {
3         println(« Test 1 ») ; // inst 1
4      }
5   else
6      {
7         if ( c )
8         {
9         println(« Test 2 ») ; // inst 2
10        }
11     }
```

How many Test cases ? To achieve a 100 % Statement coverage.

4.4 Structure-based or White-box techniques

Pour garantir une couverture totale d’instruction dans notre exemple, il nous faut 2 Cas de test :

1/ A= true ; B= false ; // ou A = false ; B= true

2/ A = false ; B = false ; c= true

4.4 Structure-based or White-box techniques

- *4.4.2 Decision testing and coverage :*

- Decision coverage, related to branch testing, is the assessment of the **percentage of decision outcomes** (e.g., the True and False options of an IF statemnt) **that have been exercised** by a test case suite.

4.4 Structure-based or White-box techniques

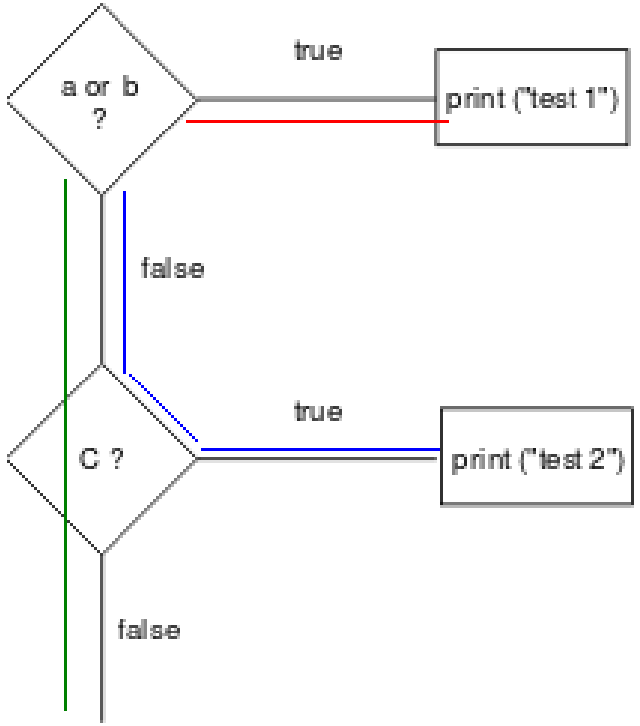
- Decision coverage is determined by the number of all decision outcomes covered by (designed or executed) test cases divided by the number of all possible decision outcomes in the code under test.
- **Decision coverage is stronger than statement coverage** 100 % decision coverage guarantees 100 % statement coverage, but not vice versa.

4.4 Structure-based or White-box techniques

```
1   if (a || b)
2      {
3         println(« Test 1 ») ; // inst 1
4      }
5   else
6      {
7         if ( c )
8         {
9         println(« Test 2 ») ; // inst 2
10        }
11     }
```

How many Test cases ? To achieve a 100 % decision/branch coverage.

4.4 Structure-based or White-box techniques



Dans notre cas, il y a trois chemins de branchement possibles pour avoir une couverture de 100 %

1/ A= true ou B= false ;
2/ A = false ; B = false ; c= true
3/ A= false; B= false ; c= false

4.4 Structure-based or White-box techniques

- *4.4.3 Other Structure-based Techniques* :

- There are stronger levels of structural coverage beyond decision coverage, for example, condition coverage and multiple condition coverage.

4.5 Experience-based techniques

- Experience-based testing is where **tests are drived from the tester's skill and intuition** and their experience with similar applications and technologies.
- A commonly used ***Experience-based technique is error guessing.***
- A **structured approach** to the error guessing tecnique is **to enumerate a list of possible defects** and to design tests that attack these defects.

4.5 Experience-based techniques

- Exploratory testing= Learn + explore + test
- Used in :

1. Agile approaches
2. With experiences tests
3. With no documentations
4. Time-pressure

4.6 Choosing Test techniques

Background :

- The choice of which test techniques to use depends on a number of factors, including the type of system, regulatory standards , customer or contractual requirements, level of risk, type of risk, test objective, documentation available, knowledge of the testers, time and budget, development life cycle, use case models and previous experience with types of defects founds.
- When creating test cases, testers generally use a combination of test techniques including process, rule and data-driven techniques to ensure adequate of the object under test.