ISTQB Training

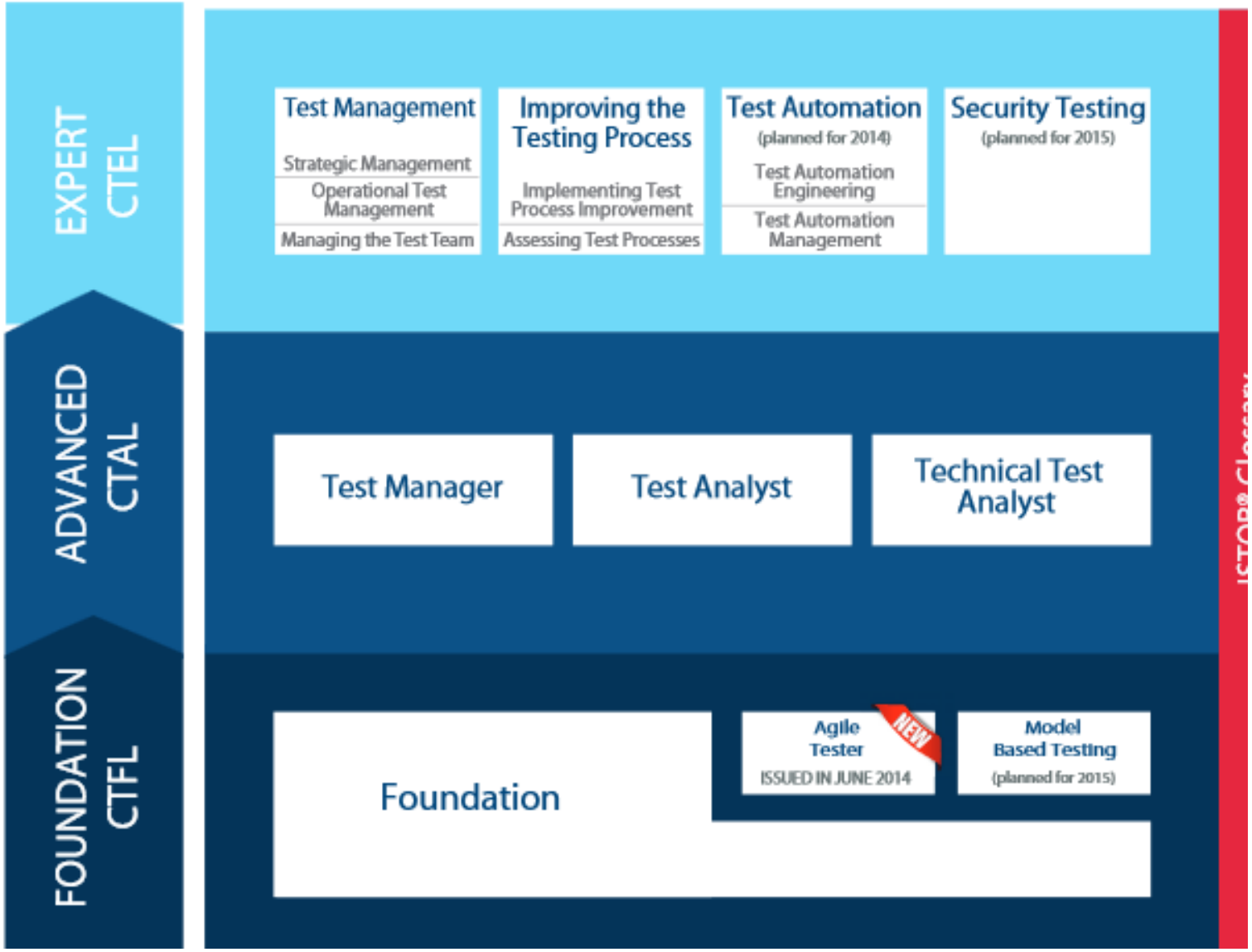
December 2016. Boughdiri Aymen

TELNET Holding

ISTQB

- Founded in Edenburg on November 2002
- It is the world's only not for profit organization dedicated solely to provide practical software tester certification
- Developped by more than 100 experts in more than 40 countries
- ISTQB certification is the most widely recognized and fastesr growing software tester certification in the world
- More than 320 000 certified testers acrros the world and over 12 000 new certificates per quarter

ISTQB


EXPERT CTEL
ADVANCED CTAL
FOUNDATION CTFL
Test Management
Strategic Management
Operational Test Management
Managing the Test Team
Improving the Testing Process
Implementing Test Process Improvement
Assessing Test Processes
Test Automation
(planned for 2014)
Test Automation Engineering
Test Automation Management
Security Testing
(planned for 2015)
Test Manager
Test Analyst
Technical Test Analyst
Foundation
Agile Tester
NEW
ISSUED IN JUNE 2014
Model Based Testing
(planned for 2015)

ISTQB

- 40 Multiple choice questions
- Each good answer gives one point
- To succeed, the participant needs to have at least 65% of good answers (26 good answers)
- The exam lasts 60 minutes

Agenda

- Chapter 1 : Fundamentals of testing
- Chapter 2 : Testing throughout the software life cycle
- Chapter 3 : Static techniques
- Chapter 4 : Test design techniques
- Chapter 5 : Test management
- Chapter 6: Tool support for testing
- Exam

Chapter 1 : Fundamentals of testing

Chapter 1 : Fundamentals of testing

- Why is testing necessary?
- What is testing?
- Testing principles
- Fundamental test process
- The psychology of testing
- Code of ethics

1.1 Why is testing necessary?

- Software systems form an integral part of our daily lives. Whether in our home or at our workplaces, software system have become essential for our comfort
- Therefore, software that does not work correctly can lead to many problems, including loss of money, time or business reputation
- For critical software, these failures can also cause major finacial losses,injury or death
- This why it is important to find and ectify the defects before it is released for a public use

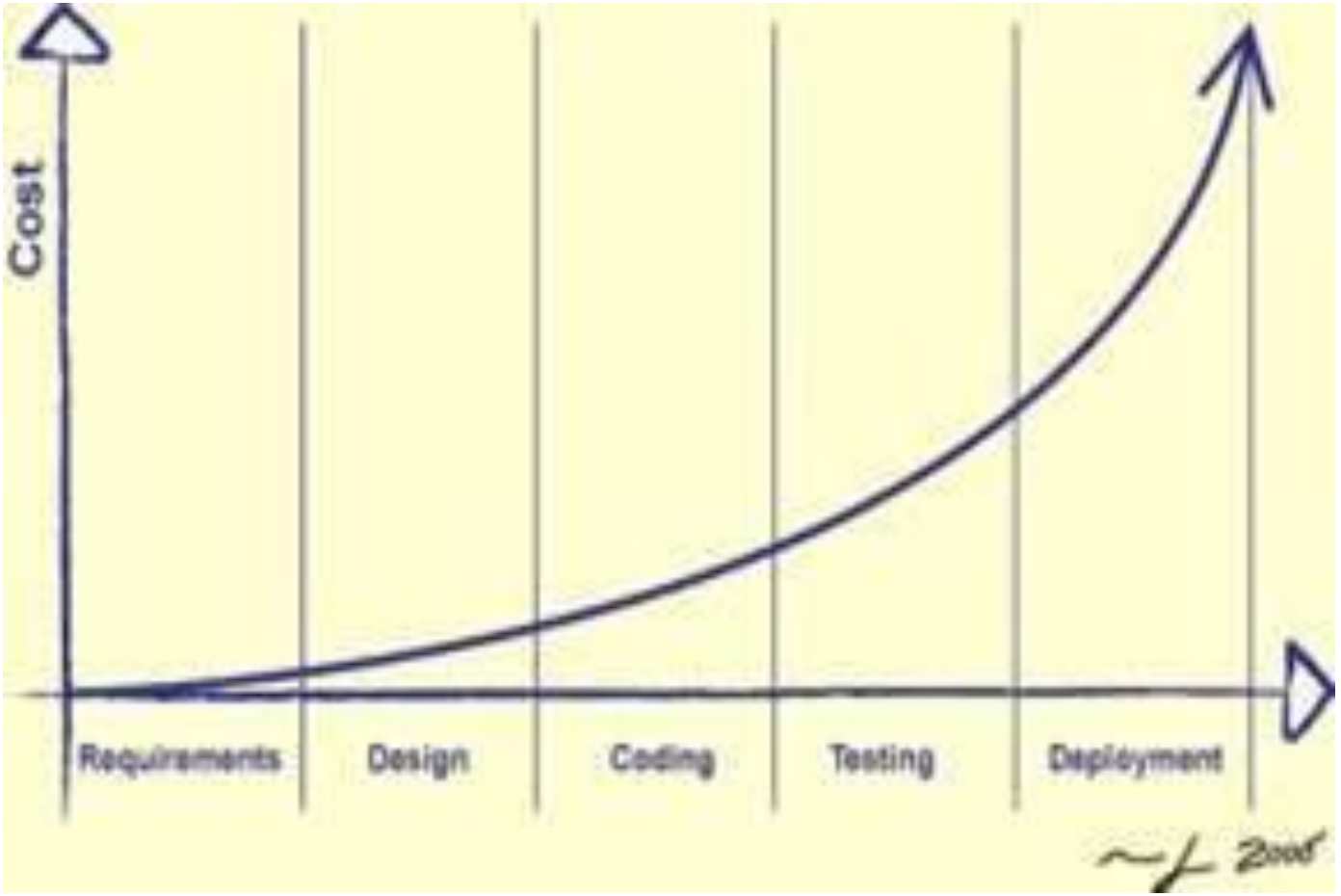
1.1 Why is testing necessary?

- 1996: Ariane 5
  - Software already used for Ariane 4 (32 bits) has been used for Ariane5 (64 bits)

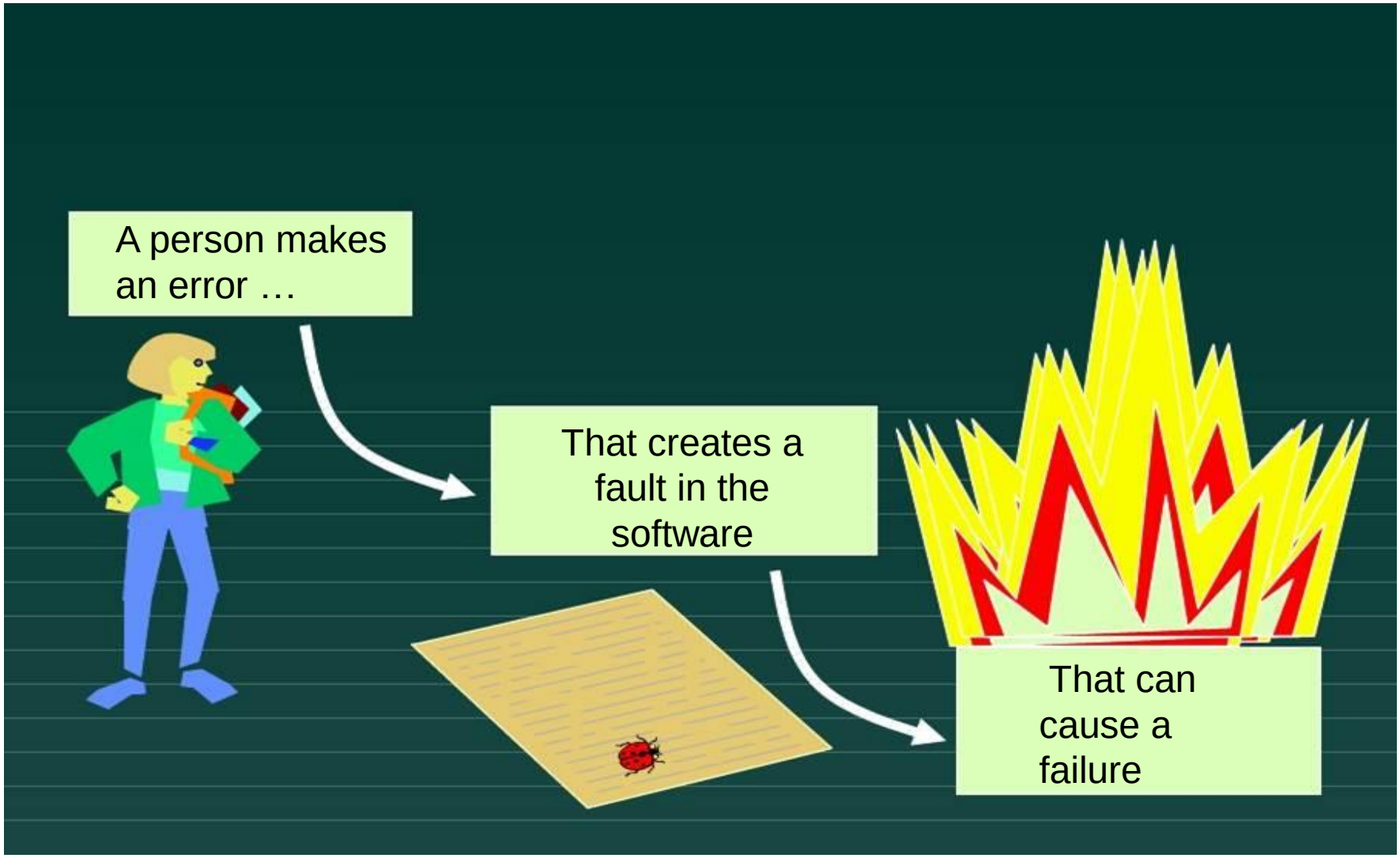
    ➔ 7 billion $
- 1999: Mars climate orbiter
  - Integration of modules using different units(Newton/Libra)

    ➔ 900 million $
- 1999: French Bank
  - Migration from windows NT to windows 2000 : incorrect software behavior

1.1 Why is testing necessary?



1.1 Why is testing necessary?



1.1 Why is testing necessary?

- Causes of software issues:
  - Error (while using the software, analyzing client requirement, designing, coding, testing …)
  - Environmental conditions (strong magnetic field, pollution ,…)

1.1 Why is testing necessary?

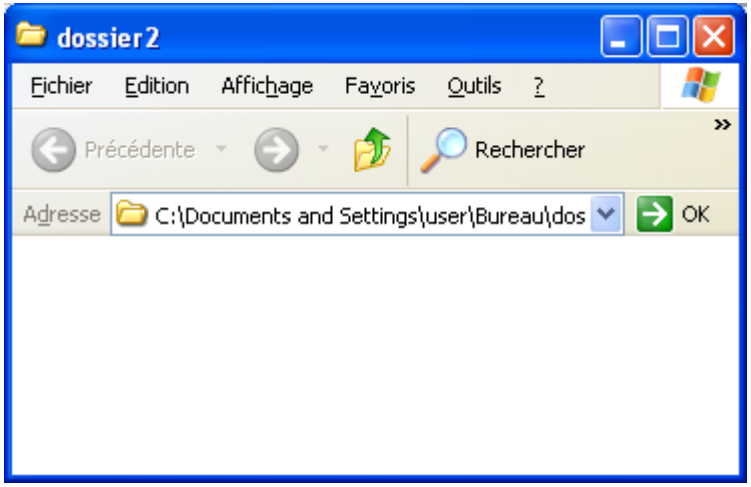
- With the help of testing, it is possible to measure the quality of the software in terms of defects found.
- By understanding the root causes of defects found in other project, processes can be improved, which in turn should prvent those defects from reoccuring and, as consequence, improve the quality of future sustems. This is an aspect of quality assurance.
- Tests is an activity of quality assurance
- Deciding how much testing is enough should take account of the **risk level**

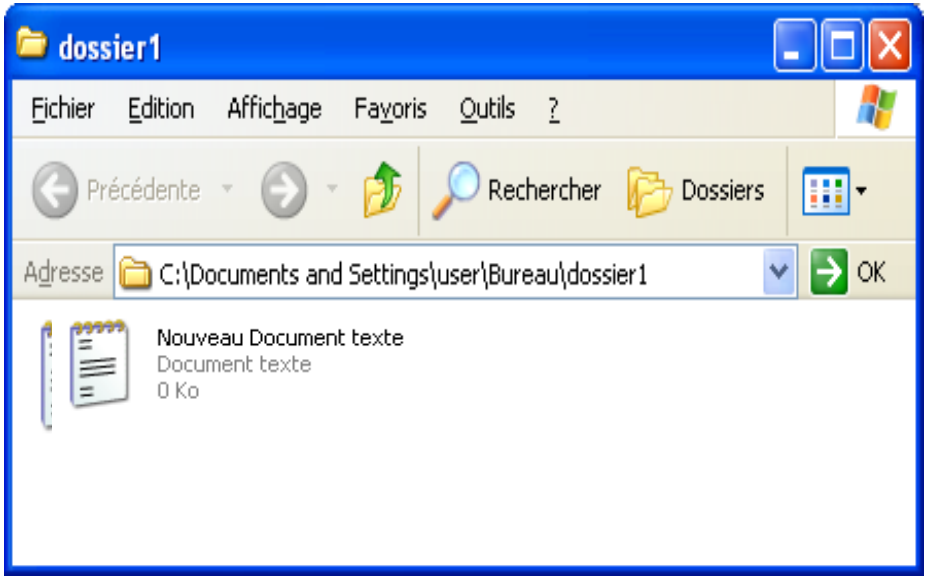
1.2 What is testing

- Test is an activity of quality assurance. It exists in all life cycle activities, both static and dynamic, concerned with planning, preparation and evaluation of software products and related work products to determine that they satisfy specified requirements, to demostrate that they are fit for purpose and to detect defects [ISTQB]

- Test objectives are:
  - **Defect detects**
  - Gaining confidence about the level of quality
  - Providing information for decision-making
  - Preventing defects

1.3 Testing principles



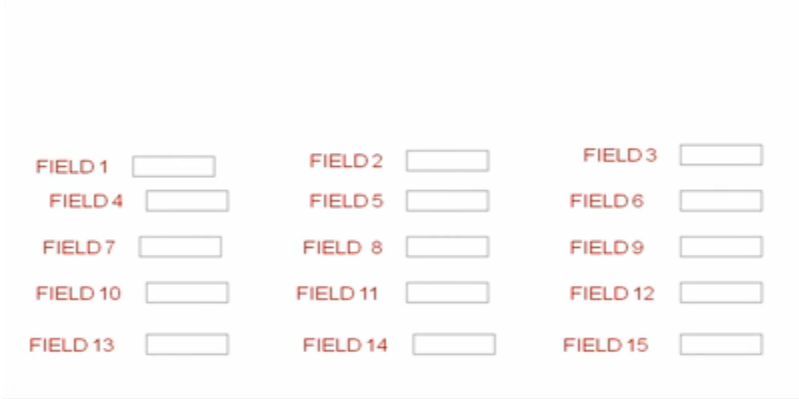


1.3 Testing principles

- Try to transfer the file when it is opened
- It has not write access to the folder B
- The folder B is shared and its carrying capacity is reached
- The file B has a file with the same name

1.3 Testing principles

- Supose we have 15 boxes to be tested, EVERY boxes have 5 values possibility. The number of test paterns is 5^15= 30517578125
- If we test all possible combinations➔ Time + cost will explose



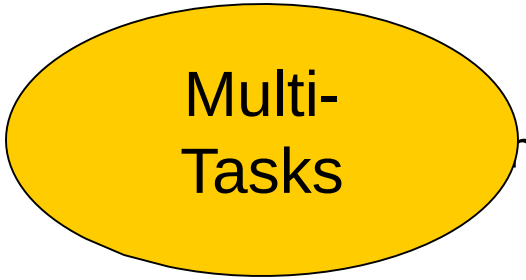
- **<u>Exhaustive testing are impossible</u>**
- Therefore we will need an optimal test sample based on the risk of the application

1.3 Testing principles

- What operation is likely to cause more product failures in your application:

  a) openMicrosoft world

  b) Open internet explorer

  c) Open 10 

me time

- **Defect clustering** : A small number of modules contains most of the defects discovered during pre-release testing

1.3 Testing principles

- If the smae tests are repeated many times, it will happen that the same set of test cases will no longer find new defects

- This is the **paradox of pesticide**

- To overcome this problem, test cases should be reviewed regulary, addong new test cases to detect new bugs

1.3 Testing principles

- We can never certify that the software does no contains any BUG

Bill Gates - Microsoft Window 98 crash on live TV.mp4

- http://www.youtube.com/watch?v=f-1TOeHY7as

- **Testing shows presence of defects**: Tests reduce the probability of undiscovered defects remaining in the software but , even if no defects are found, it is not a proof of correcness

1.3 Testing principles

» If we certify that our softawre does not contain a BUG at 99%, but this software does not satisfy client needs

» **<u>Abscence of error fallacy</u>** : Find and fixing defects does not help if the system built is unusable and does not fulfill the user's need and expectations

1.3 Testing principles

- To fix this issue, there is the princile: **Early testing**

- Test activities shall be started as early as possible on the software system development life cycle

- **Testing is context dependent** : Testing is done differentlly in different contexts

1.3 Testing principles

- Principe 1 – testing shows presence of defects
- Principe 2 – Exhaustive testing is impossible
- Principe 3 – Early testing
- Principe 4 – Defect clustering
- Principe 5 – Pesticide paradox
- Principe 6 – Testing is context dependent
- Principe 7 – Abscence of errors fallacy

1.4 Fundamental Test Process

1.4 Fundamental Test Process

» **Test planning and control :**

– Test planning is:

- Define the objectives of tests
- Define test approach
- Define the needed ressources (humain; Materials (PC), test environment …)
- Define the exist criteria

– The control is the ongoing activity of comparing actul progress against the plan, and reporting the status, including deviation from the plan. It involves taking actions necessary to meet the mission and the objectives of the project.

1.4 Fundamental Test Process

- **Test analysis and design :**
  - Test analisis and design is the activity during wich the general test objectives are transformed into tangible test conditions and test cases
  - The test analysis and design actively has the following major tasks:
    - Reviwing the test basis (such as requirement, software integrity level (risk level), risk analisys reports, architecture, design, interfacespecification)
    - Evaluating testability of test basis and test objects

1.4 Fundamental Test Process

- Designing and prioritizing high level test cases
- Designing the test environment setup and identifying any requirement infrastructure and tools

1.4 Fundamental Test Process

» **Test implementation and execution :**

– Test implementation and execution has the following major tasks:

- Finalyzing, implementation and prioritizing test cases
- Developping and prioritizing test procedures, creating test data
- Creating test suites from thet test procedures forefficient test execution
- Verifying that test environment has been setup correctly
- Execute test procedures according to the planned sequence

1.4 Fundamental Test Process

- Loggingthe outcome of test execution and recording the identities and versions of the software under test, test tools and tesware
- Comparing actual results with the expected results
- Reporting discrepancies as incidents and analyzing them in order to establish their cause

1.4 Fundamental Test Process

» **Evaluating exist criteria and reporting :**

– Evaluationg exist criteria is the activity where test execution is assessed against the defined objectives

– Evaluationg exist criteria has the following major tasks:

- Cecking the test logs against the exist criteria specified in test planning
- Assessing if more tests are needed or if the exist criteria specified should be changed
- Whriting a test summary report for stakeholders

1.4 Fundamental Test Process

» **Test closure activities :**

– Test closure activities include the following major tasks:

- Checking which planned delivrables have been delivred
- Closing incident report or raising change records for any that remain open
- Documenting the acceptance of the system
- Finalyzing and archiving testware, the test environement and the infrastructure for later reuse
- Handing aver the testware to the maintenance organization

1.4 Fundamental Test Process

- Analyzing lessons learned to determine changes needed for future releases and projects
- Using the information gathered to improve test maturity

1.5 The psychology of testing

- A certain degree of independence often makes the tester more effective at finding defects and failures
- Several levels of independance can be defined as showen here from low to high:
  - Tests designed bty the person who wrote the software under test
  - Tests designed by another person (e.g from the developement team)
  - Tests designed by a person from a different organizational group (e.g independent test team)
  - Tests designed by a person from a diffrent organization or company (outsourcing or certification by an external body)

1.5 The psychology of testing

- Remind every one of the comon goal of better quality systems
- Consider all stakeholders as one team
  - Favor frequent communication
  - Informal meetings between developers and testers
- Try always to justify error
  - We all make errors
  - Explain that by knowing about this now we can work round it

1.6 Code ethics

- Involvement in software testing enables individuals to learn confidential and privileged information. A code of ethics is necessary, among other reasons to ensure that the information is not put to inappropriate use.

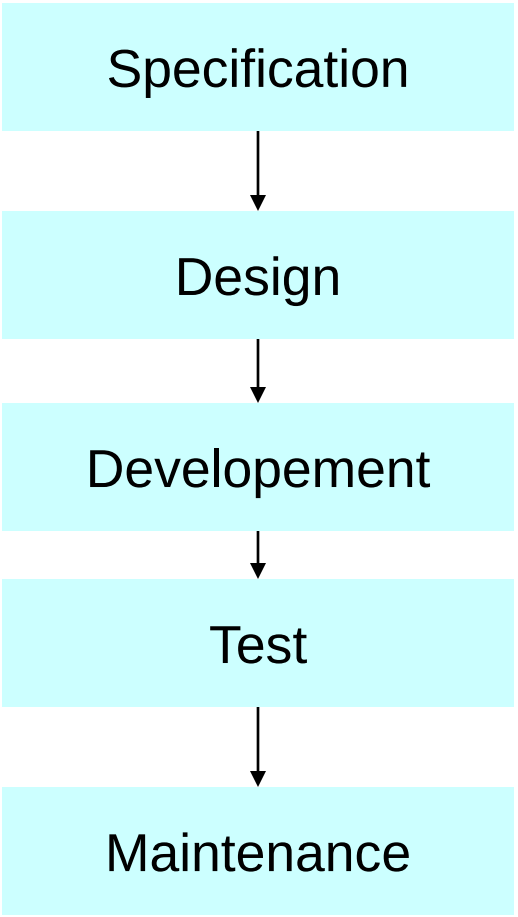
Chapter 2 : Testing throughout the software life cycle

Chapter 2 : Testing throughout the software life cycle

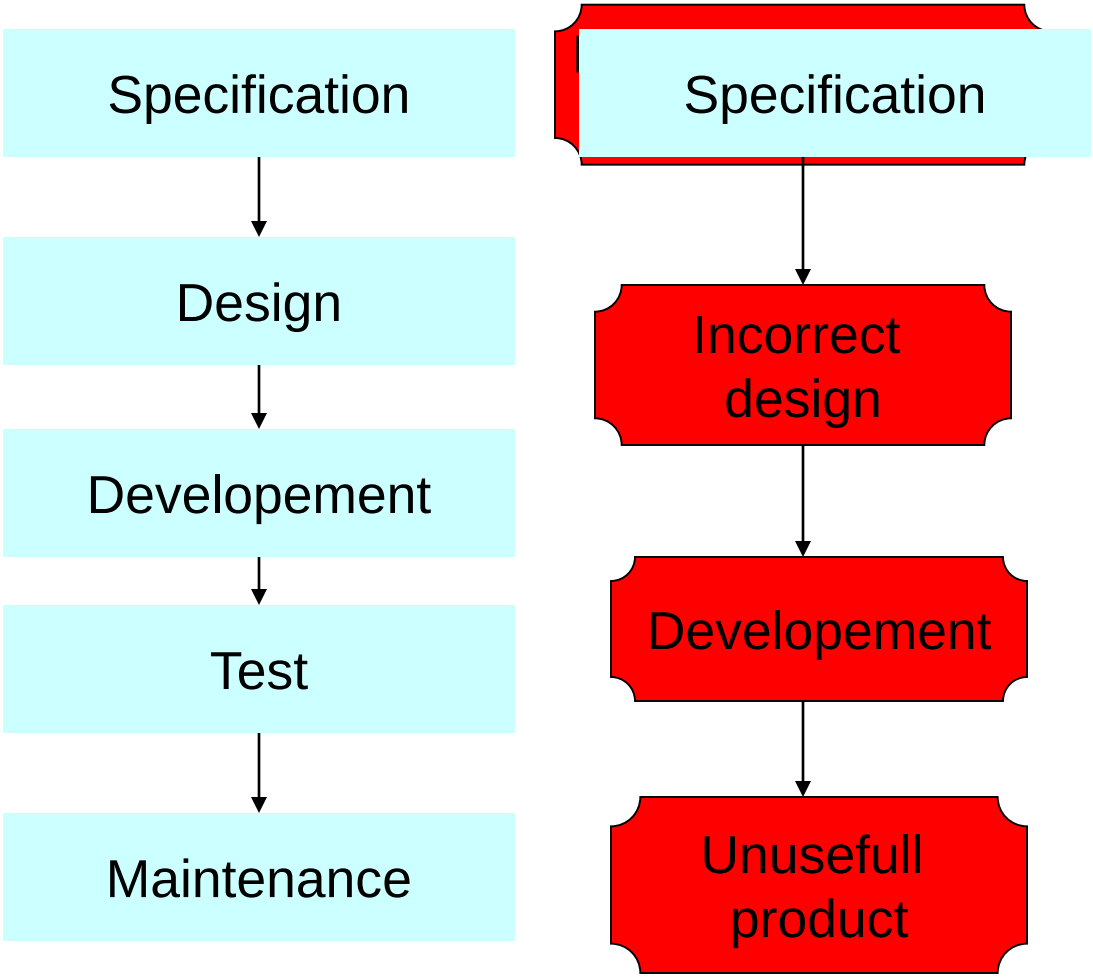
- Software development models
- Test levels
- Test types
- Maintenance testing

2.1 Waterfall model

Waterfall model

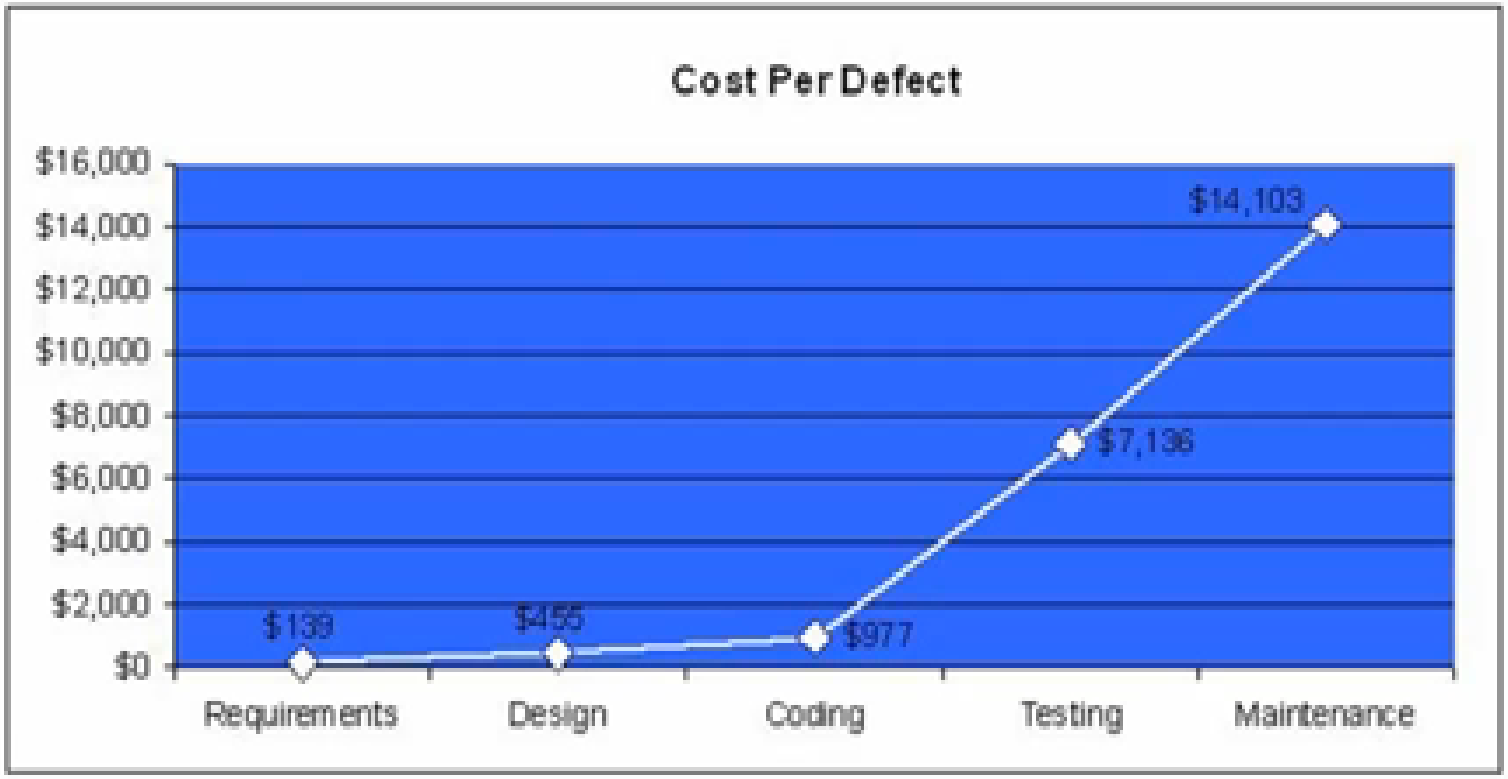


2.1 Waterfall model

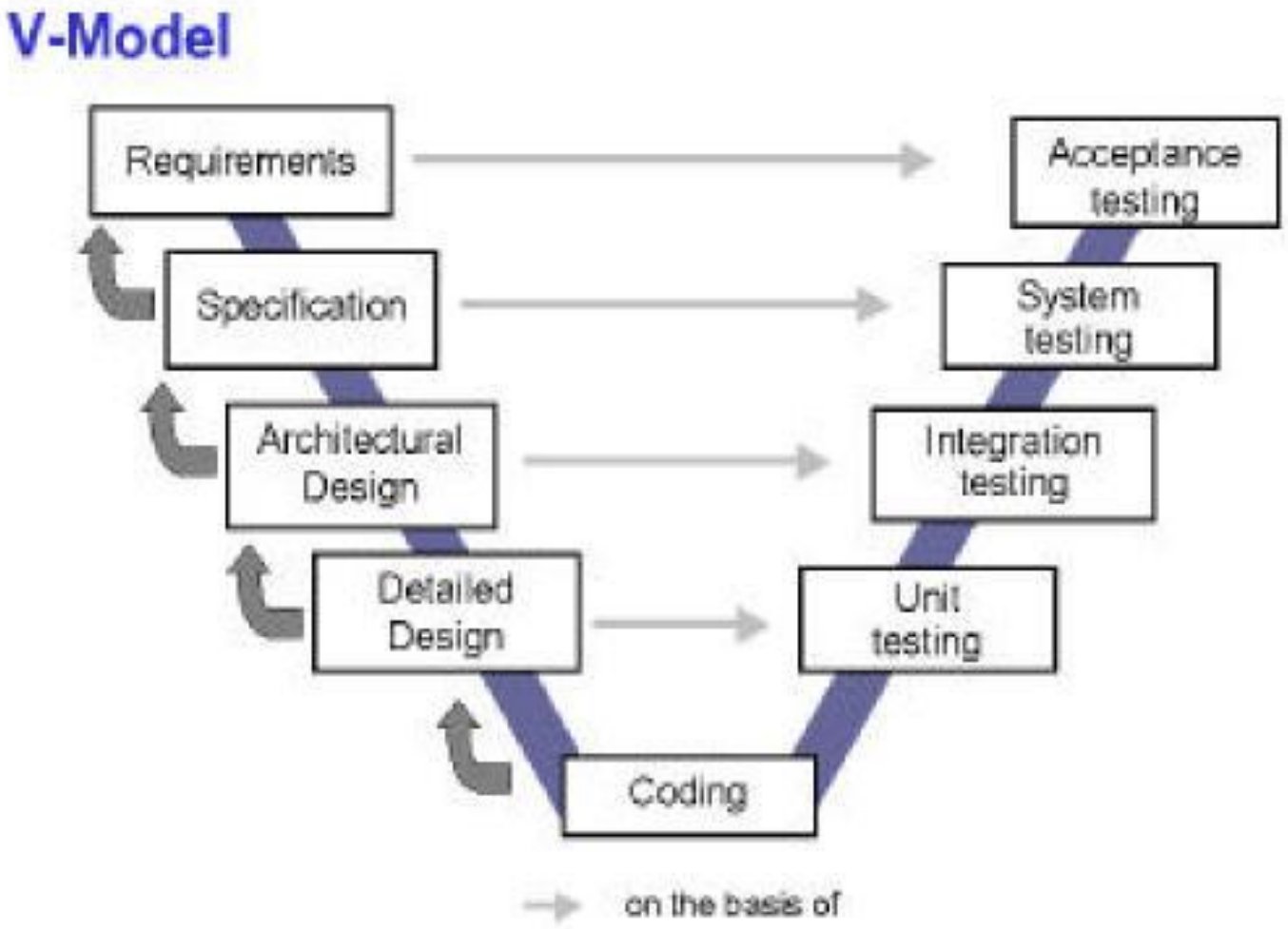


2.1 Waterfall model

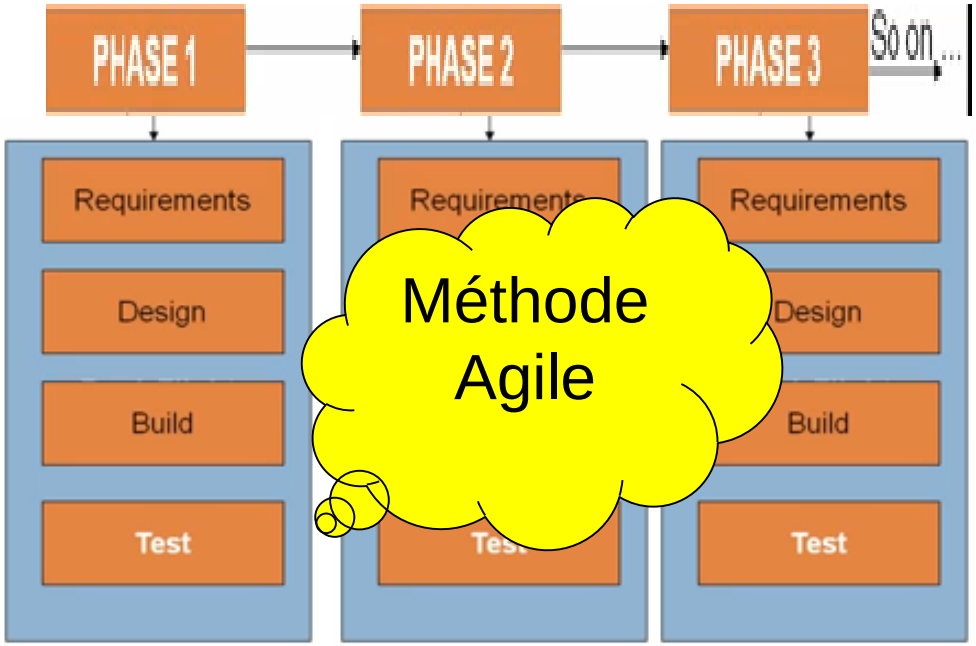
50% of failures are introduced at the customer requirements collection and writing specification



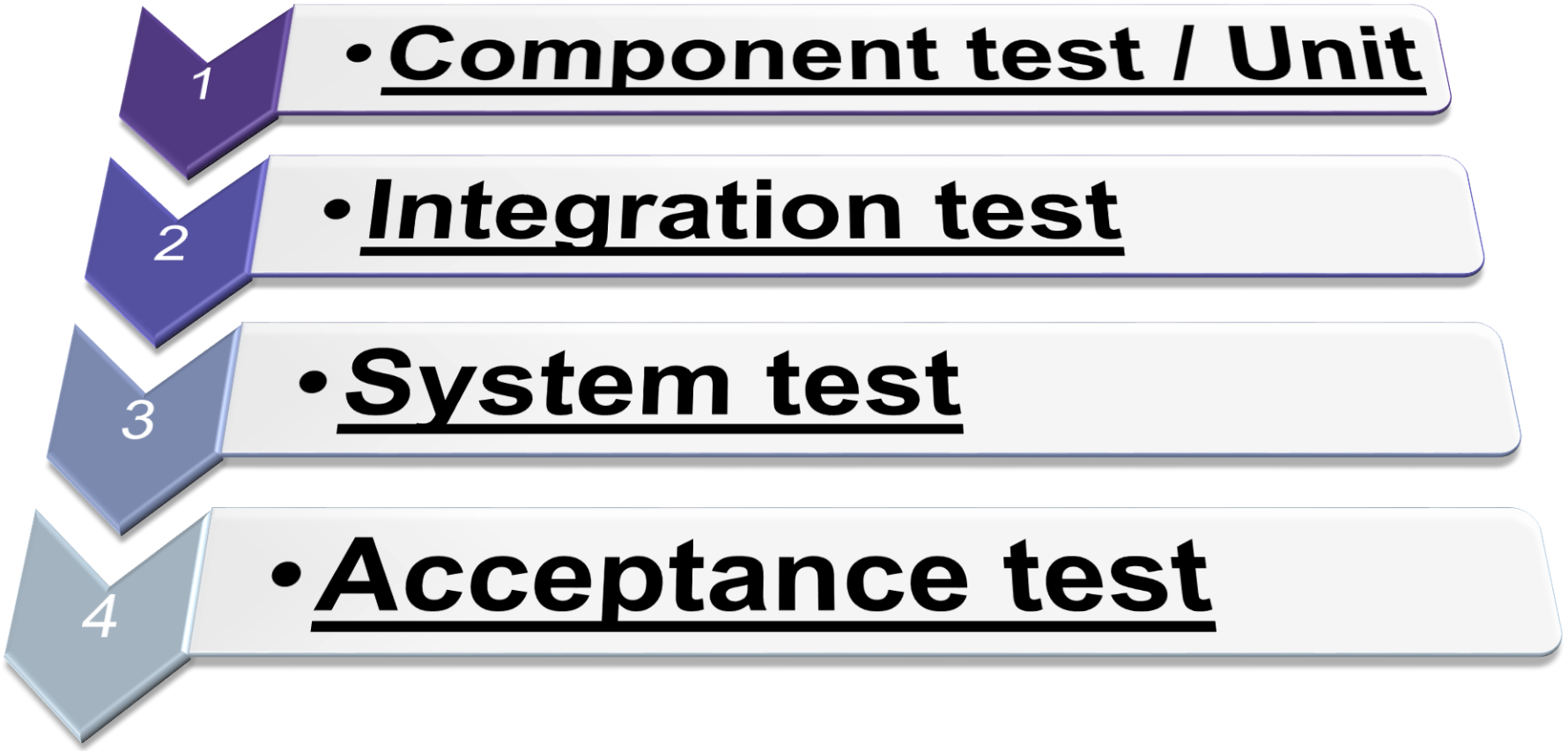
2.1 V Model


V-Model
Requirements
Specification
Architectural Design
Detailed Design
Coding
Acceptance testing
System testing
Integration testing
Unit testing
on the basis of

Modèle itératif



2.2 Test levels



2.2 Test levels

- **Component (Unit) testing :**
  - Searches for defects and verifies the functioning of modules, programs, objects, classes that are separately testable
  - It may include testing of functionality and specific non functional charcteristics
  - Typically, componement testing occurs with the access to the code being tested and with the support ofthe developement environment(unit test framework, debugging tool …)
  - One approach to componement testing is to prepare and automate test cases before coding, called test-driven developement

2.2 Test levels

- **Integration test:**
  - Component integration testing tests interactions between software componement and it is done after component testing
  - System integration testing tests the interaction between different systems and my be done after system testing
  - At each stage of integartion, testers concentrate solely on the integration itself and not the functionality of modules
  - Ideally, testers should understand system's architecture and integration strategies

2.2 Test levels

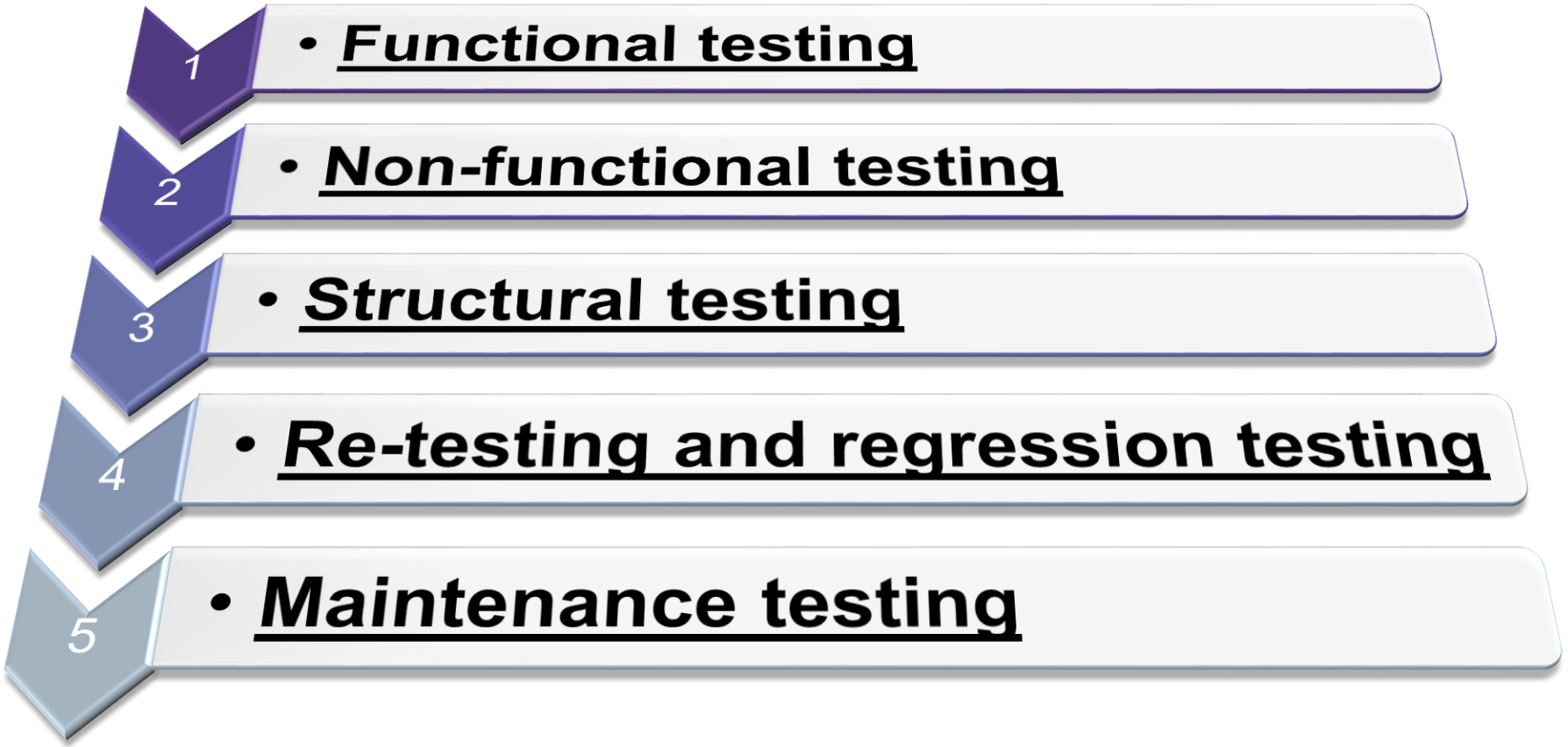
- **System testing:**
  - Concerned with the behavior of a whole system/product as defined by the scope of a developement project
  - The test environement should correspond to the final target or production environement as much as possible in order to minimize the risk of environement specific failures not found in testing
  - System testing should investigate both functional and non-functional requirements system
  - Based on specifications

2.2 Test levels

» **Acceptance testing:**

- It is often the responsability of the customers or users of the system
- The goal is to establish confidence in the system
- La recherche d'anomalies n'est pas l'objectif principal des tests d'acceptation.
- Typical forms of acceptace testing include:
  - User acceptance test
  - Operation acceptance test (system administrators: backuping.security ….)
  - Contract and regular acceptance testing
  - Alpha (internal) and beta (external) testing

2.3 Test types



2.3 Test types

- **Functional testing :**
  - Functional tests are based on functions and features and may performed at all test levels
  - It is testing of ‘What’ the system does
- **Non-Functional testing:**
  - It includes but not limited to performance testing, load testing, stress testing, usability testing and maintenability testing
  - It is testing of ‘How’ the system works
  - It is testing the caracteristics of te system

2.3 Test types

- **Structural testing (white box) :**
  - Used to mesure the thoroughness of testing through assessment of coverage of a type of structure
  - Structural testing mauy be performed at all test levels but espatially in componement testing and componement integration testing
- **Re-testing (confirmation testing:**
  - To confirm that the original defect has been successfully removed
  - Debugging (defect fixing) is a developement activity not testing activity

2.3 Test types

- **Regression testing :**
  - It is the repeated testing of an already tested program, after the modification to discover any defect introduced or uncovered as a result of the change(s).
  - It may performed at all test levels and applies to functional, non-functional and structural testing
  - Regression tests are run many timeq and generally envolve slowly. It is a strong candidate for automation

2.3 Test types

» **Maintenance testing :**

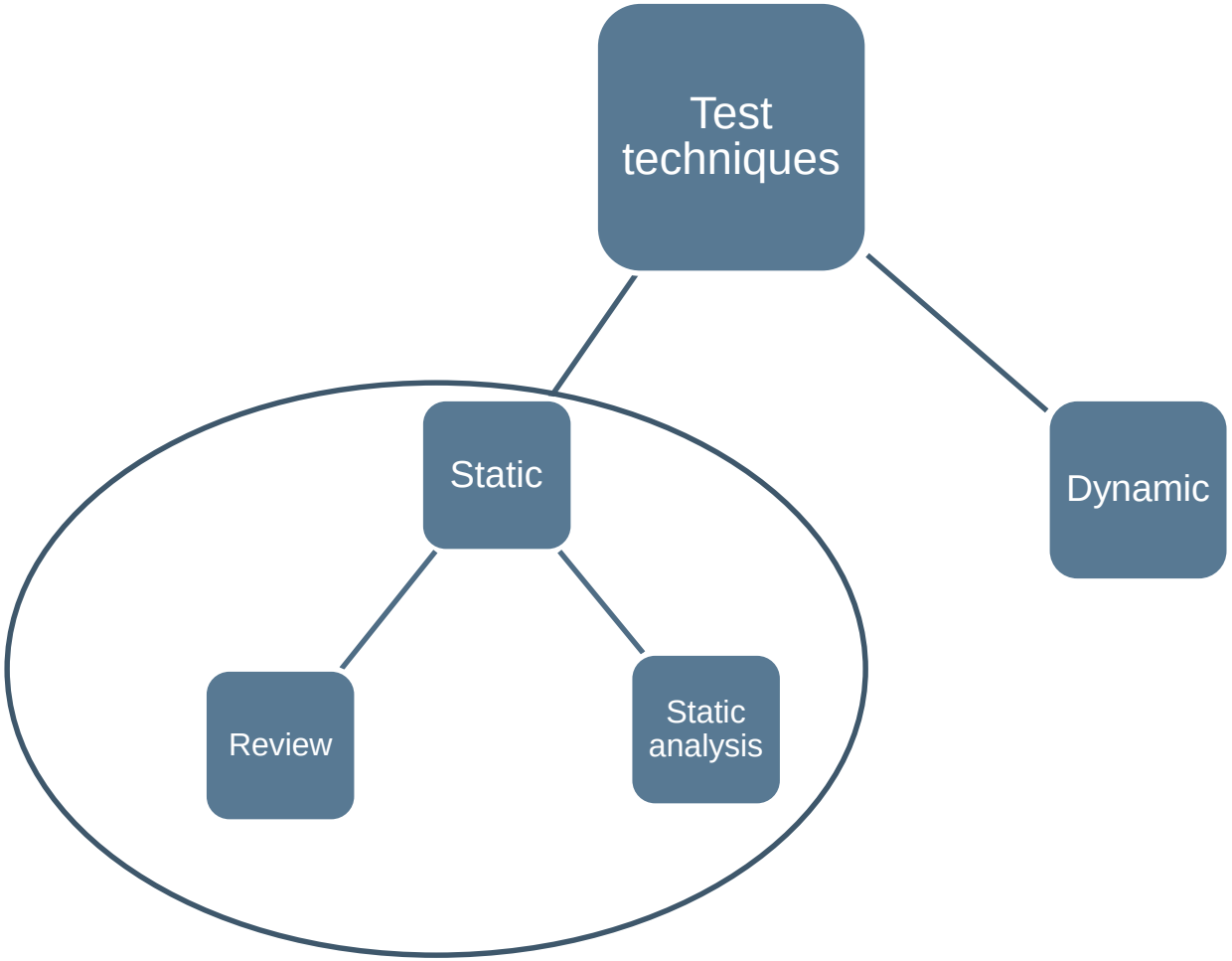
- It is performed on a deployed system due to a change or extention of the system or its environement
- Usually maintenance testing will consist of two parts:
  - Testing the changes
  - Regression tests to show that the rest of the system has not been affected by the maintenance work
- A major and important activity within maintenance testing is impact analysis

Chapter 3 : Static techniques

Chapter 3 : Static techniques

- Static testing
- Review process
- Static analysis by tools

3.1 Test techniques



3.1 Static techniques and testing process

- The Static Testing Techniques are based on the manual examination (reviews) or the code analysis (static analysis) of project documentation without running the code
- All software product can be reviewed , including requirements specifications, design specifications, code, test plans, test specifications, test cases, test scripts , user guides or web pages
- A review could be done entirely manually or by using the support tools
- static techniques find causes of defects , the dynamic tests are failures themselves

3.2 Review process

- **Review phases**
- Planning:
  - Define the review criteria , entry and exit criteria for more formal review types and select the part of the documents to be reviewed
  - Selecting the personnel and allocating roles
- Kick-off:
  - Distribute documents and explain the objectives, process ...
- Individual preparation:
  - Review documents, writing of potential defects , questions and comments …

3.2 Review process

- Examination/evaluation/recording of result
  - Discuss or record , with documented results or minutes (for more formal review types)
  - Note the defects, making recommendations regarding the treatment of defects , making decisions about defects
- Rework
  - Fixing defect found (typically done by the author)
  - Recording updated ststus of defect

3.2 Review process

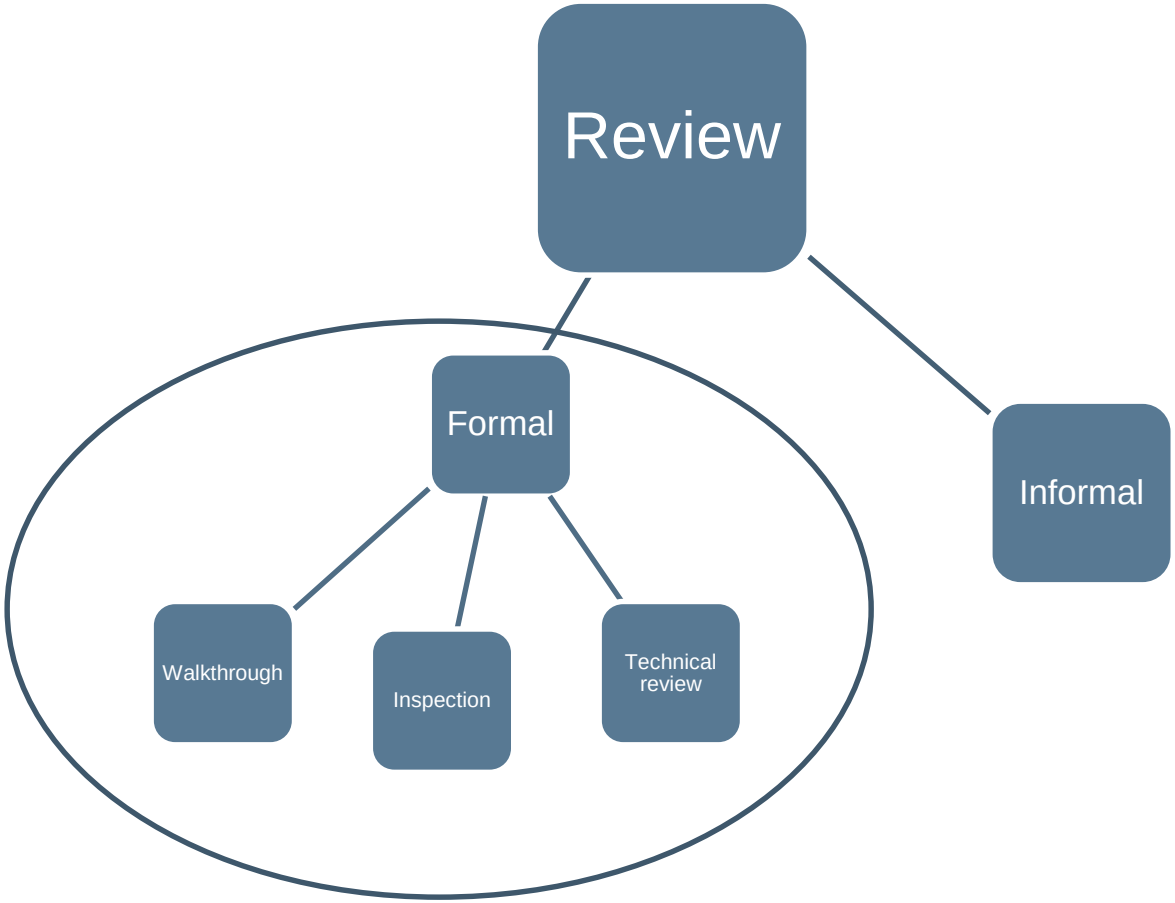
- Follow-up:
  - Checking that the defects have been adressed
  - Gathering metrics
  - Cheking on exist criteria

3.2 Review process

» Role and responsabilities

- Manager: decides the execution reviews, allocates time in project planning and determines if the review objectives have been met
- Moderator: the person who leads the review of documents , including planning and implementation of the review, and following-up after the meeting .
- Author : The author or person with chief responsability for the document to be reviewed
- Reviewers: individuals with a specific business or technical background who identify and describe findings in the product under review
- Scribe: documents all the issues , problems and open points identified during the meeting

3.2 Types of review



3.2 Review process

» Review types

- Informal review
  - No formal process
  - May include pair programming or design review and code by a technical manager
  - The results can be documented.
  - Main purpose: Easy Way to get the results

3.2 Review process

- Technical review
  - A peer group discussion activity that focuses on achieving consensus on the technical approach to be taken
  - The goal of the technical review are to:
    - Assess the value of technical concepts and alternatives in the product and project environement
    - Establish consistency in the use and reprensentation of technical concepts
    - Ensure at an early stage that technical concepts are used correctly

3.2 Review process

- Walkthrough
  - A step-by-step presentation by the author of a document in order to gather information and to establish a comon understanding od its content
  - The goal of a walkthrough are to:
    - Present the document to stakeholders both within and outside the software decipline in order to gather information
    - Explain and evaluate the contents of the document
    - Examine and discuss teh validity of proposed solutions and the viability of alternatives, establish consensus

3.2 Review process

- Inspection
  - A type of peer reviw that relies on visual examination of documents to detect defects, e.g. violations of developement standards and non-conformance to higher level documentation
  - The goals of inspections are:
    - Help the author to improve the quality of the document
    - Remove defects efficiently as early as possible
    - Create a comon understanding by exchanging information among the inspection participants
    - Learn from defects found to improve processes

3.2 Review process

- **How we can succeed a review?**
  - Pick things that really count
  - Explicity plan and track review activities
  - Train participants
  - Just do it!
  - Report result
  - Continuously improve process and tools

3.3 Static analysis by tools

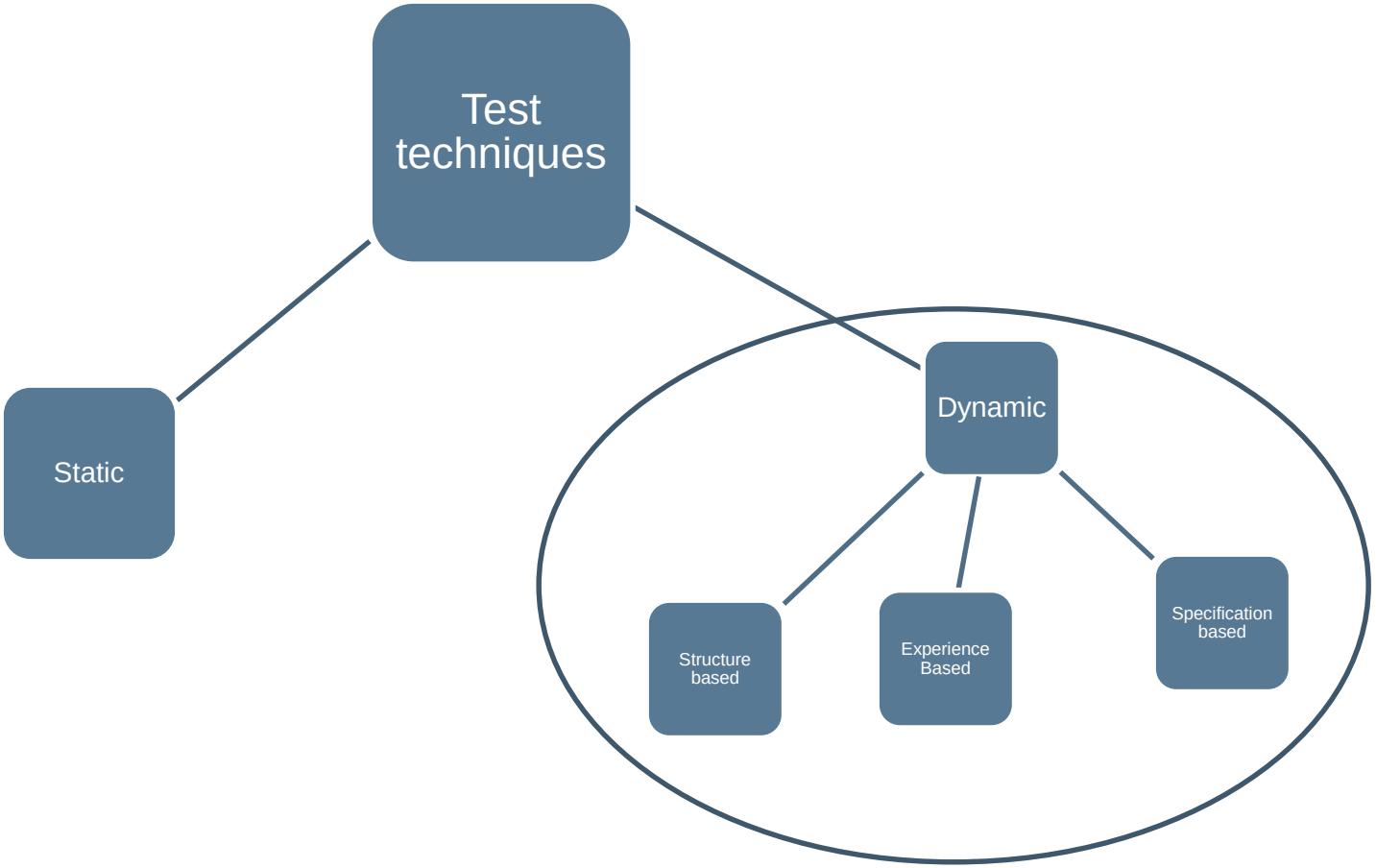
- Static analisys differs from more traditional dynamic testing in a mumber of imporatant ways:
  - Static analysisis ideally performed before the types of a formal review
  - Static analisys is unrelated to dynamic properties of the requirements, design and code, such a test coverage
  - The goal of the static analisys is to find defects, whether or not they may cause failures
- As with the review, static analysis finds defects rather than failures

3.3 Static analysis by tools

- The most common static analysis features in day-to-date practice are:
  - Coding standards
  - Code metrics
  - Code structure

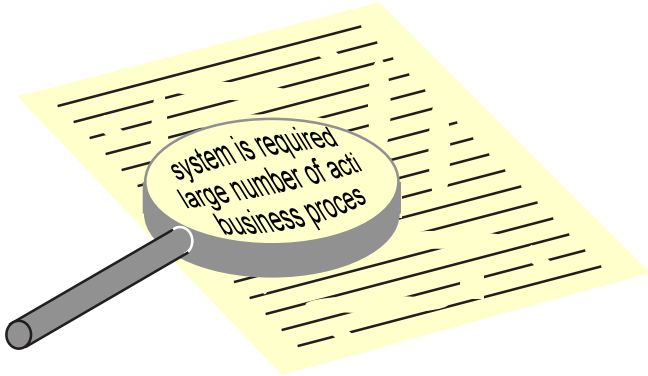
Chapter 4 : Test design techniques

4.1 Test design techniques



4.2 Categories of test design techniques

- Static (without execution)
  - Examination of documents , source code ...



- Functional (black box)
  - Based on the system behavior (specification)

- Structure (white box)
  - Based on structure

4.2 Categories of test design techniques

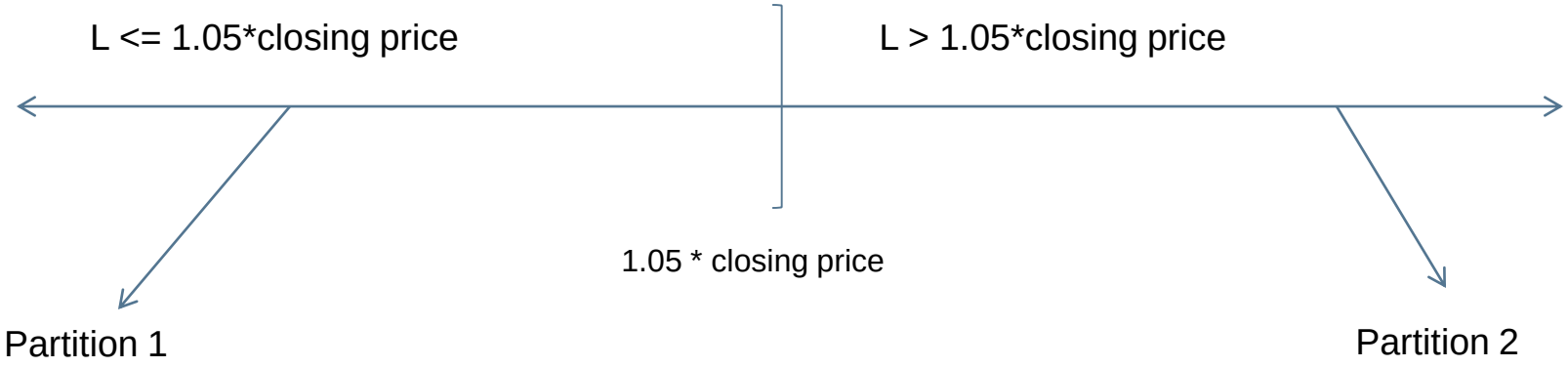
- Black box design techniques (techniques based on the specifications) are a way to derive and select test conditions , test cases or the test data
- Black box design techniques include functional testing and non-functional testing
- Techniques based on the specification does not use any information concerning the internal structure of a component or system to be tested
- White box design techniques (techniques based on structures) are based on an analysis of the component or the system structure
- These two techniques can be combined with techniques based on experience

4.3 Black box techniques

- **Equivalence partitioning:**
  - The inputs of software are divided into groups that need to have a similar behavior and identical treatment.
  - Tests can be designed to cover all partitions
  - Partitions are applicable to all test levels

4.3 Black box techniques

- Closing price filter 5%
  - If Limit <= 1.05 * closing price , Order acknowleged
  - If Limit > 1.05 * closing proce , Order rejected

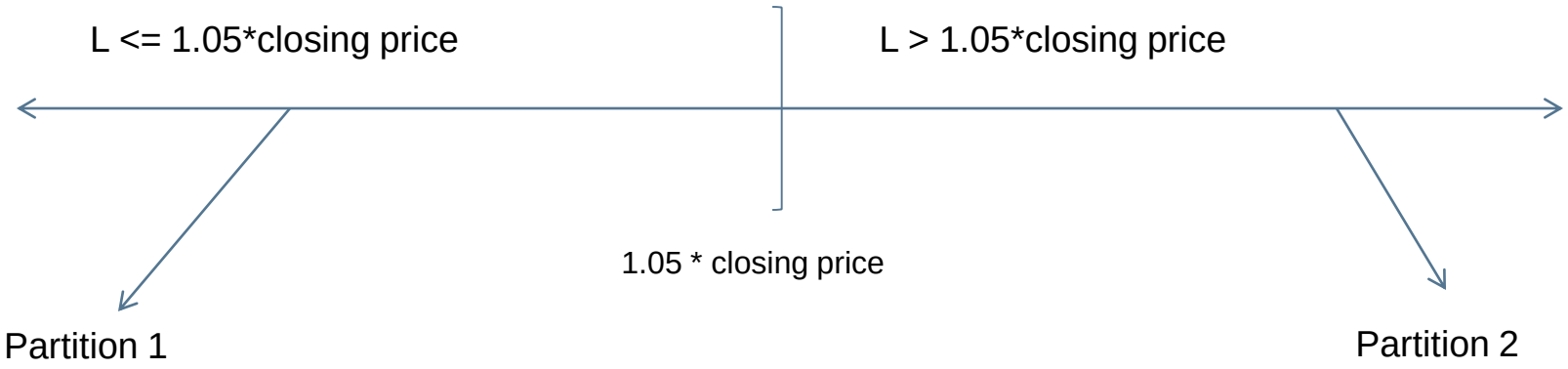


4.3 Black box techniques

- **Boundary value testing**
  - The min and max values of a partition are its limits.
  - Tests can be designed to cover the limits valid and invalid
  - A value of each limit is selected for a test
  - The analysis of these values can be applied to all test levels
  - It is a complementary technique to that of equivalence partitions

4.3 Black box techniques

- Closing price filter 5%
  - If Limit <= 1.05 * closing price , Order acknowleged
  - If Limit > 1.05 * closing proce , Order rejected



BV1= 1.05*closing price

BV2= 1.05*closing price -tick

4.3 Black box techniques

» **Exercice**

Invalid partition	Valid (for 3% interest)		Valid (for 5%)		Valid (for 7%)
–$0.01	$0.00	$100.00	$100.01	$999.99	$1000.00

Test conditions	Valid partitions	Invalid partitions	Valid boundaries	Invalid boundaries
Balance in aU$_o$Unt	$0.00 $100.00	< $0.00	$0.00	-$0.01
	$100.01-$999.99	>$Max	$100.00	$Max+0.01
	$1000.00- $Max	non-integer (if balance is an input field)	$100.01	
			$999.99	
			$1000.00	
			$Max	

4.3 Black box techniques

- **Decision table**
  - A black box test design technique in which test cases are designed to execute the combinations of inputs and/or causes shown in decision table

4.3 Black box techniques

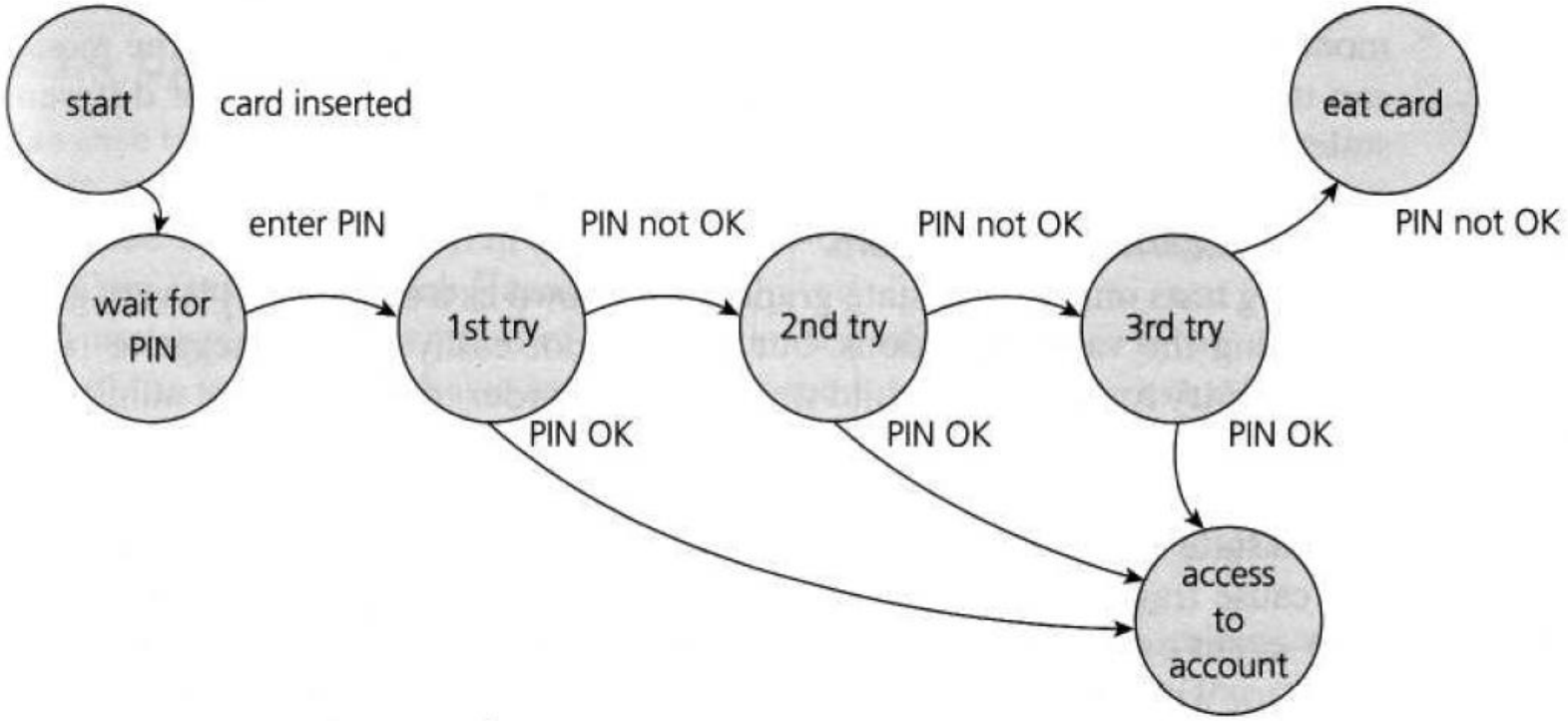
Conditions	Rule 1	Rule 2	Rule 3	Rule 4
Modality	Market	Market	Any Price	Any Price
Validity	FOK	E&E	FOK	E&E
Actions				
Results	Y	Y	N	N

4.3 Black box techniques

- **State transistion testing**
  - A black box test design technique in which test cases are designed to execute valid and invalid state transition
  - This aspect of the system can be shown by a diagram of states and transitions
  - This allows the tester to view the software in terms of states , transitions between states , data entries and actions that may result from these transitions

4.3 Black box techniques

» **Example:**



4.3 Black box techniques

- **Use case**
  - A use case describes interactions between actors and systems
  - Each use case has preconditions that must be reached to execute the successfully the case
  - Each use case terminates with post-conditions , which are the observable results and final state of the system after the end of the execution of the use case

4.3 Black box techniques

» **Example:**

	Step	Description
Main Success Scenario **A: Actor** **S: System**	1	A: Inserts card
	2	S: Validates card and asks for PIN
	3	A: Enters PIN
	4	S: Validates PIN
	5	S: Allows access to account
Extensions	2a	Card not valid S: Display message and reject card
	4a	PIN not valid S: Display message and ask for re-try (twice)
	4b	PIN invalid 3 times S: Eat card and exit

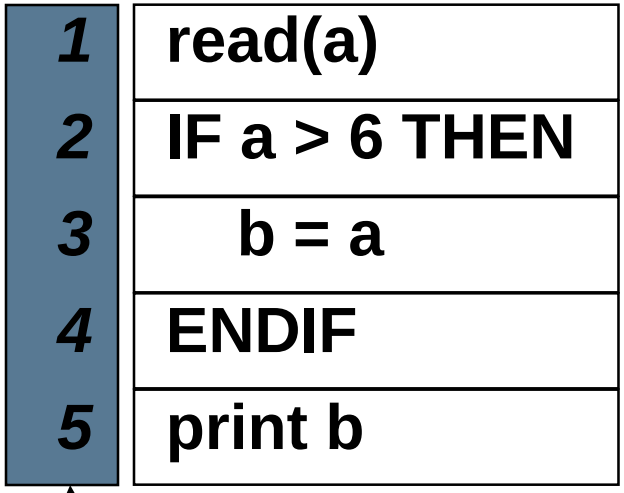
4.4 Structural techniques

» **Statement coverage :**

– Statement coverage is the assessment of the percentage of executable statements that have been executed by a test case suite

$$= \frac{\text{Number of statements (can be executed)}}{\text{Total number of statement (can be executed)}}$$

4.4 Structural techniques



Test case	Input	Expected output
1	7	7

Statement numbers

As all 5 statements are ‘covered’ by this test case, we have achieved 100% statement coverage

4.4 Structural techniques

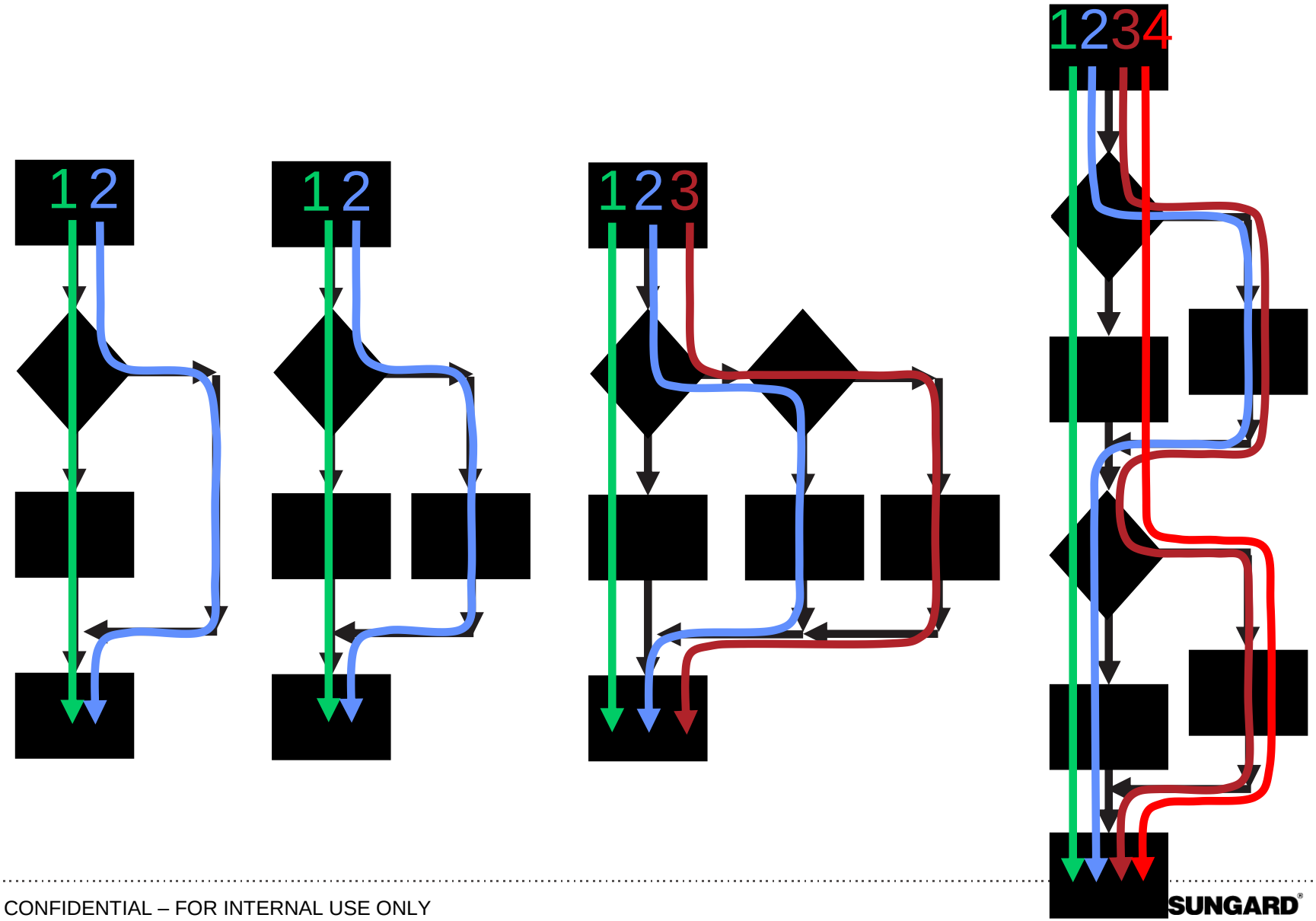
- **Test des décisions et couverture:**

- Decision coverage , related to branch testing, is the assessment of results of decisions percentages (eg the True and False options of an IF statement) that have been processed by a series of test cases

- Decision coverage is greater than the statement coverage ::
  - A 100% coverage decisions ensures 100% coverage instructions.
  - A 100% coverage instructions does not ensure 100% coverage decisions

$$= \frac{\text{number of decisions outcomes exercised}}{\text{total number of decision outcomes}}$$

Path coverage



Example 1

Wait for card to be inserted

IF card is a valid card THEN

display “Enter PIN number”

IF PIN is valid THEN

select transaction

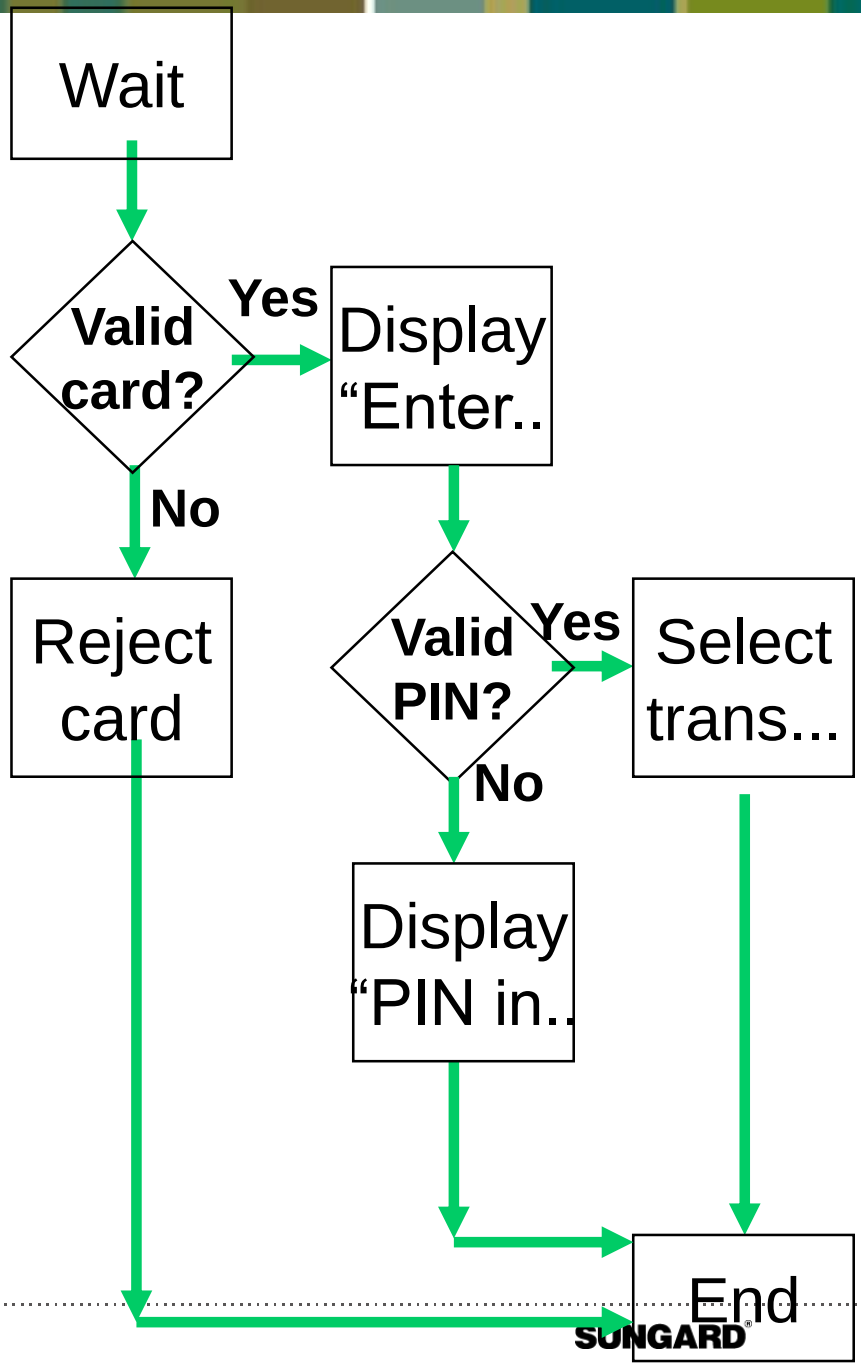
ELSE (otherwise)

display “PIN invalid”

ELSE (otherwise)

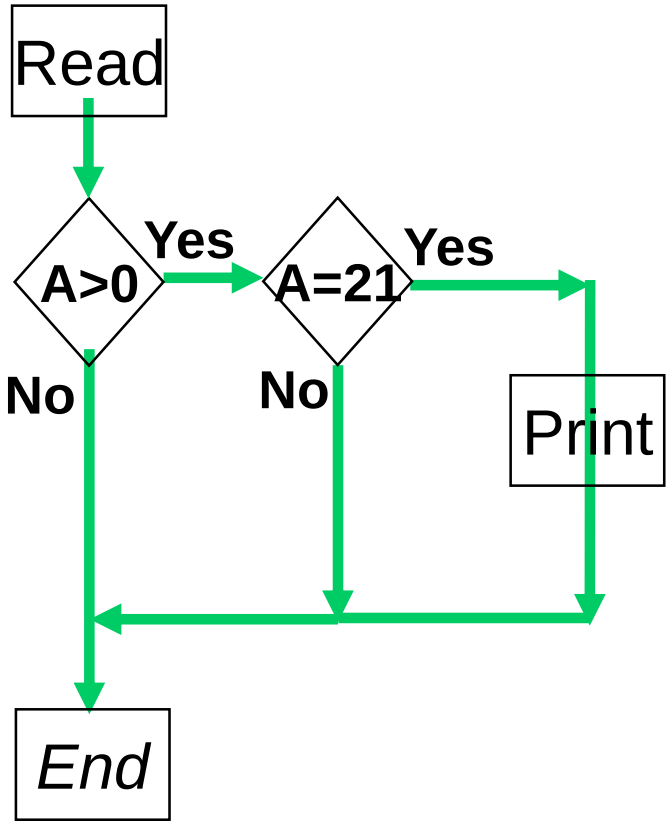
reject card

End



Example 2

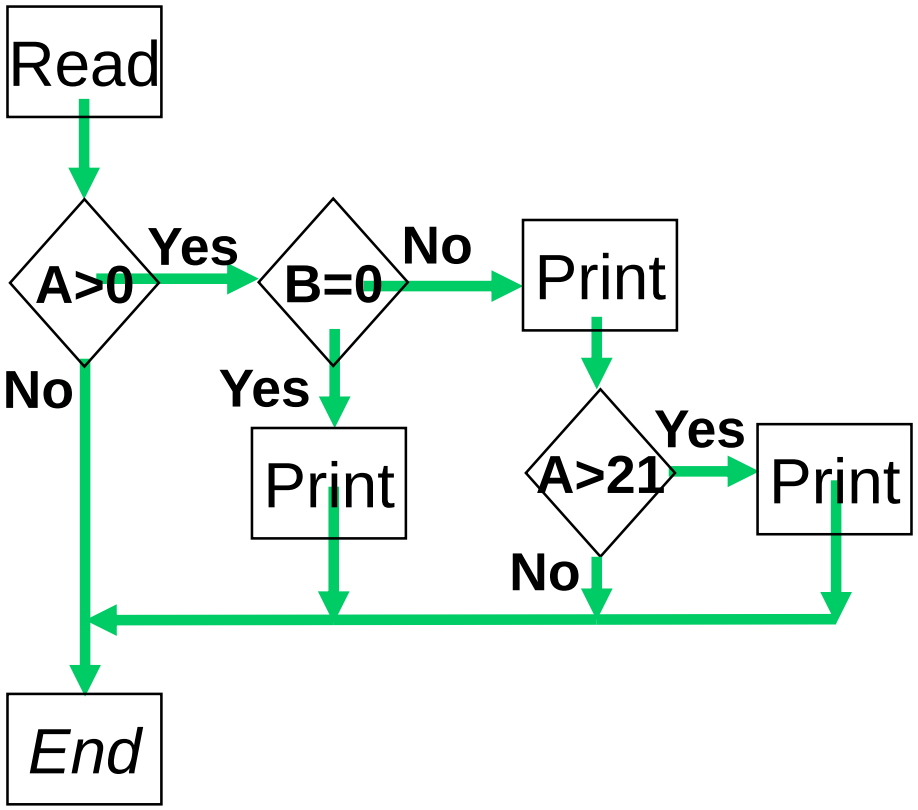
```
Read A
IF A > 0 THEN
    IF A  = 21 THEN
        Print “Key”
    ENDIF
ENDIF
```



- Cyclomatic complexity: 3
- Minimum tests to achieve:
  - Statement coverage: 1
  - Branch coverage: 3

Example 3

```
Read A
Read B
IF A > 0 THEN
    IF B  = 0 THEN
        Print “No values”
    ELSE
        Print B
        IF A > 21 THEN
            Print A
        ENDIF
    ENDIF
ENDIF
```

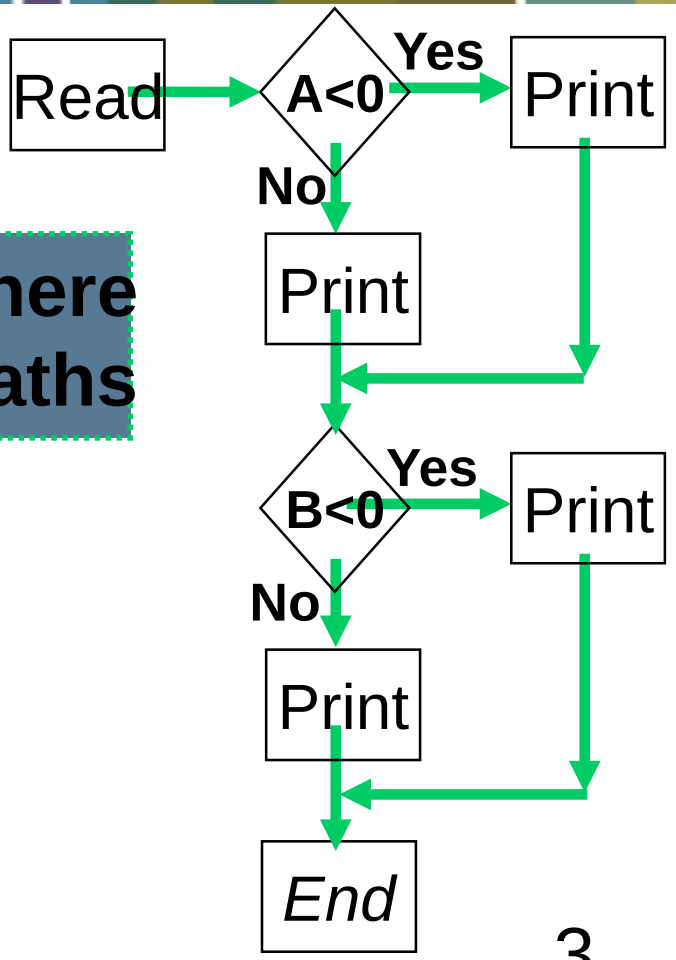


- Cyclomatic complexity: 4
- Minimum tests to achieve:
  - Statement coverage: 2
  - Branch coverage: 4

Example 4

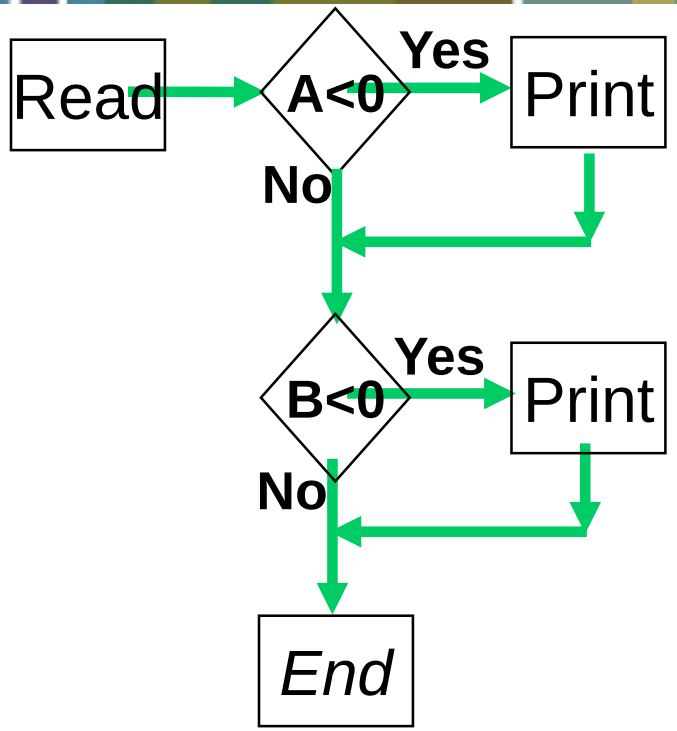
```
Read A
Read B
IF A < 0 THEN
    Print  “A negative”
ELSE
    Print  “A positive”
ENDIF
IF B < 0 THEN
    Print  “B negative”
ELSE
    Print  “B positive”
ENDIF
```

Note: there are 4 paths



- Cyclomatic complexity: 3
- Minimum tests to achieve:
  - Statement coverage: 2
  - Branch coverage: 2

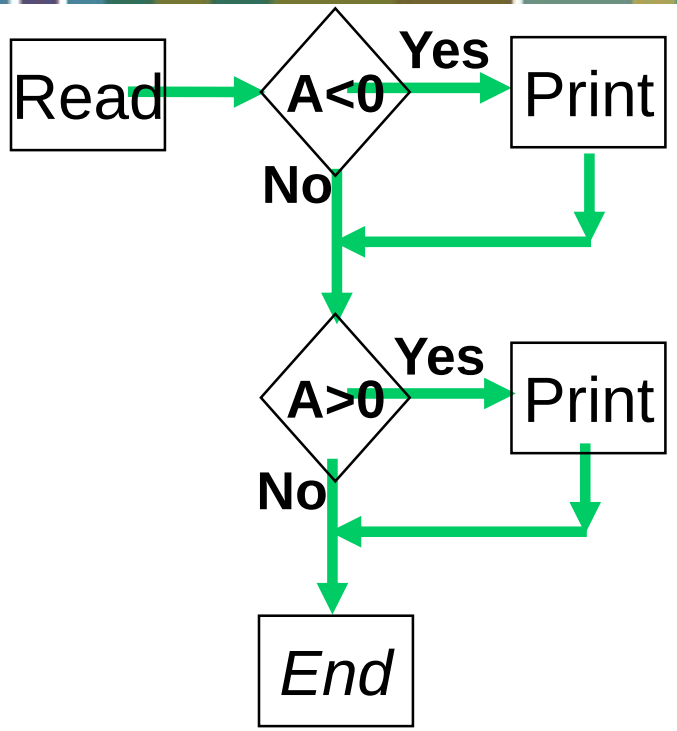
Example 5




```
Read A
Read B
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF B < 0 THEN
    Print  “B negative”
ENDIF
```

- Cyclomatic complexity: 3
- Minimum tests to achieve:
  - Statement coverage: 1
  - Branch coverage: 2

Example 6




```
Read A
IF A < 0 THEN
    Print  “A negative”
ENDIF
IF A > 0 THEN
    Print  “A positive”
ENDIF
```

- Cyclomatic complexity: 3
- Minimum tests to achieve:
  - Statement coverage: 2
  - Branch coverage: 2

4.4 Structural techniques

» Condition coverage

- The concept of coverage can also be applied to other levels of testing
- For example, the integration level , the percentage of modules , components or classes that have been executed by a series of test cases can be expressed as a module coverage, components or classes...

4.5 Experienced based testing

- **Error guessing:**
  - Depe,dent of the skill of the tester
  - No rules, think of situation in which the software may not be able to handle
  - May be used as a complement to other formal techniques
- **Exploratory testing**:
  - Usefull when there are no or poor specifications and when time is severely limited
  - The tester is constantly making decissions about what to test next and where to spend the limited time

4.6 Choosing a test techniques

- There is no best
- There are diffrents point to consider when choosing the test techniques:
  - System to be tested
  - Goal of testing
  - Client requirements
  - Risk
- When creating test cases , testers generally use a combination of testing techniques

Chapter 5 : Test management

5.1.Test management

- Test organization
- Test plans, estimates and strategies
- Test progress monitoring and control
- Configuration management
- Risk and testing
- Incident management

5.1.Test management

» Test organization: Independent or integrated testing?

- The benefits of independence:
  - Independent testers see the various defects and of a different nature and are impartial
  - An independent tester can verify the assumptions made during the specification and implementation of the syste
- Drawbacks of independent testing:
  - Tester become isolated
  - Developpers lose the responsability of quality

5.1. Test organization

» Tasks of test leader and a tester

– Typical test leader tasks may include:

- Write or review a test strategy for the project
- Plan tests
- Initiate the specification, preparation, implementation and execution of tests, monitor the test results and check the exist criteria
- Adapt planning based on test results and progress and take any action necessary to compensate for problems
- Introduce suitble metrics for measuring progress and evaluation the quality of testing and the product

5.1. Test organization

- **Typically tester tasks may include**
  - Review and contribute to test plan
  - Analyze, review and asses user requiremenetw, specifications and models for testability
  - Create test specifications
  - Set up the test environment
  - Prepare and acquire test data
  - Implement tests, execute and log the tests, evaluate the results and document the deviations from expected results
  - Review tests developped by other

5.2 Test planning and estimation

- Test plan is a project plan for the testing work to be done

- Why do we write a test plans?
  - Writing a test paln guides our thinking
  - A test plan help us to manage change
  - A test plan serve as vehicles for communicating whith other memebers of the project team and other stakeholders

5.2 Test planning and estimation

» **Estimating what testing will involve**

– We may estimate testing effort by:

  • Consulting the people who will do the work and other people whith expertise on the tasks to be done

  • Analyze metrics from past projects

  • Use a sophisticated approaches such as TAP (test point analysis)

– There are three factors that might impact the test effort:

  • Product (project documentation, complexity, …)

  • Process (availability of test tools, the life cycle adopted, ressources,…)

  • The result of testing

5.3 Test progress monitoring and control

- Test monitoring can serve various purpose during the project, including the following:
  - Give the test team and the test manager feedbacks on how the testing work is going
  - Provid ethe project team with visibility about the test results
  - Measure the status of teh testing, test coverage and test items against the exist criteria to determine whether the test work is done
  - Gather data for use in estimating future test efforts
- Test reporting is often about enlightening and influencing project stakeholders by analyzing the information and metrics available to support conclusion, recomendation and decission
- Test control is about guiding and corrective actions to try to acheive the best possible the outcome for the project

5.4 Configuration management

- The purpose of the configuration management is to establish and maintain the integrity of the products of the software or system through the project and product life cycle

- The test configuration manaement may involves:

  - All items are identified, version controlled, tracked for changes, related to each other and related to developement ites so that traceability can be maintained throughout the test process

  - All identified documents and software items are refrenced unambiguously in test documentation

- For the tester, configuration management helps to uniquely identify (and reproduce) the tested item, test dociments, the tests and the test harness

5.5 Risk and testing

» Project risk

- Organizational factors : lack of competence and Staff training, personal problems , Policy Issues , ...
- Technical problems : Environment unavailable test, quality of design , code and test data
- Problems of acquisition : On failure a third party, contractual problems , ...

» Product risk

- Risks related to the product are a particular type of risk to the success of a project
- The test, as risk control activity provides feedback about the residual risk by measuring the removal efficiency of critical defects and contingency plans
- Risks are used to decide when to start testing and where to test more; The test is used to reduce the risk that an adverse event occurs or to reduce the impact of the latter
- In addition , the test can help identify new risks , determine what risks should be minimized and reduce uncertainty about risks

5.6 Incident management

- The discrepancies between actual and expected outcomes need to be logged as an incident
- An incident must be analyzed and mau turn out to be a defect
- In order to manage all incidents to completion, an organization should establish an incidents management process and rules classification
- Incidents may be raised during developement, review, testing or use of a software product
- Incidents reports have the following objectives:
  - Provide a developpers and other parties with feedback about the problem to enable identification, isoolation and correction if necessary
  - Provide test leaders a means of trackingthe quality of the system under test and the progress of the testing

Chapter 6 : Tool support for testing

6.1. Tool support for testing

- Types of test tool
- Effective use of test tool
- Introducing a tool into an organization

6.1 Types of test tool

- Tool for support management of testing and tests
  - Test management tools
  - Requirement management tool
  - Incident management tool
  - Configuration management tool

6.1 Test management tool

Product	Company	Category
Quality center	HP	Paid
TestManager	IBM	Paid
CARS	Compuware	Paid
Testlink	X	Free
Salome	X	Free
Xstudio	X	Free

6.1 Requirement management tool

Product	Company	Category
Quality center	HP	Paid
Doors	IBM	Paid
Caliber RM	Borland	Paid
Testlink	X	Free
Salome	X	Free
Xstudio	X	Free

6.1 Incident management tool

Product	Company	Category
Quality center	HP	Paid
ClearQuest	IBM	Paid
TrackRecord	Borland	Paid
Mantis	X	Free
Xstudio	X	Free

6.1 Configuration management tools

Product	Company	Category
VSS	Microsoft	Paid
Clearcase	IBM	Paid
Perforce	Perforce software	Paid
CVS	X	Free
Subversion	X	Free

6.1 Test tool execution

Product	Company	Category
QTP	HP	Paid
Rational Robot	IBM	Paid
Test partner	Compuware Paid	Paid
Badboy	X	Free
Selenium	X	Free

6.2 Effective use of the tool

- Potential benefits of using the tools
  - Reduce of the repetitive works
  - Greater consistency and repeatability
  - Objective assessment
  - Ease of access to information about tests or testing
- Risk of using tools
  - Unrealistic expectation for the tool
  - Underestimating the time, cost and effort for initial introduction of the tool
  - Underestimating the effort reqired to mainain the test assets generated by the tool
  - Over-reliance on the tool

6.3 Introduire un outil dans une organisation

» Les principes

- Evaluation de la maturité de l'organisation, de ses forces et de ses faiblesses et identification des possibilités d'amélioration du processus de test.
- Evaluation au regard d'exigences claires et de critères objectifs.
- Une preuve de concept (test d'évaluation du bon fonctionnement de l'outil avec l'existant en terme d'infrastructure et avec l'outil sujet de test).
- Evaluation du vendeur (aspects de formation, de support et de commerce).
- Identification des exigences internes pour le soutien et la tutelle dans l'utilisation de l'outil
- Evaluation de besoin de formation.
- Evaluation du rapport coût/bénéfice basés sur un cas métier concret

6.3 Introducing a tool in an organization

- Assessment of the organization’s maturity
- Identification ofthe area whithin the organization where tool support will help to improve process
- Evaluating of tools against clear objective criteria
- POC
- Identifying and planning internal implementation

Test

- To start **ISTQB Mock Test - 1** , Click Here
- To start **ISTQB Mock Test - 2** , Click Here
- To start **ISTQB Mock Test - 3** , Click Here
- http://www.testingexcellence.com/istqb-quiz/istqb-foundation-practice-exam-1/