# TESTES E PERFORMANCE WEB

## TECNOLOGIAS E SISTEMAS DE INFORMAÇÃO PARA A WEB

**POLITÉCNICO DO PORTO**
ESCOLA SUPERIOR DE MEDIA ARTES E DESIGN

# M07 – INTRODUCTION TO TESTS

TSIW 2023/2024

## AGENDA

1. Fundamentals of testing;
2. Testing throughout the software development life cycle;
3. Test techniques;
4. Test management.

# FUNDAMENTALS OF TESTING

# 1. FUNDAMENTALS OF TESTING

- **Motivation;**
- What is testing?
- Why is testing necessary?
- Seven testing principles;
- Test process.

# 1. FUNDAMENTALS OF TESTING

## Motivation

- In the software world, the technological triumph has not been perfect;
- Almost every living person uses information technologies, and most of us have dealt with the frustration and wasted time that occurs when software fails and exhibits unexpected behaviours;
- Some unfortunate individuals and companies have experienced financial loss or damage to their personal or business reputation as a result of defective software;
- A highly unlucky few have even been injured or killed by software failures;
- One way to overcome such problems is software testing, when it is done well.

# 1. FUNDAMENTALS OF TESTING

## Motivation

*Wired News*'s 10 Worst Bugs (9)

**June 4, 1996:** The European Ariane 5 rocket reused code from the earlier Ariane 4 rocket. The Ariane 4 includes code that converts a 64-bit floating-point number to a 16-bit signed integer.

On its first flight, the Ariane 5's faster engines cause the 64-bit numbers to be larger than in the Ariane 4, triggering an overflow condition that results in the flight computer crashing.

# 1. FUNDAMENTALS OF TESTING

## Motivation

### Therac-25 causes radiation overdoses

The Therac-25 was a radiation therapy machine manufactured by AECL in the 80s, which offered a revolutionary dual treatment mode. It was also designed from the outset to use software based safety systems rather than hardware controls.

The removal of these hardware safety measures had tragic consequences, as race conditions in the codebase led to the death of three patients, and caused debilitating injuries to at least three other patients. The manufacturer ultimately became the target of several lawsuits from families of the victims, and became subject to a Class I recall from the FDA, a situation which only happens if the agency believes there is significant risk of death or serious injury through continued use of a medical device.

# 1. FUNDAMENTALS OF TESTING

- Motivation;
- **What is testing?**
- Why is testing necessary?
- Seven testing principles;
- Test process.

# 1. FUNDAMENTALS OF TESTING

## What is testing?

- Testing is more than running tests:
  - A misperception about testing is that it only involves running tests;
  - Carrying out a sequence of actions on the system under test, submitting various inputs along the way and evaluating the observed results is only one element of software testing;
  - There are many other activities involved in testing process;
  - There are major test activities both before and after test execution such as test planning, analyzing, designing, implementing and reporting.

# 1. FUNDAMENTALS OF TESTING

## What is testing?

- Testing is more than verification:
  - Another misperception about testing is that it is only about checking correctness; that is, that the system corresponds to its requirements, user stories, or other specifications;
  - Checking against a specification (**verification**) is certainly part of testing, but just conforming to a specification is not sufficient testing;
  - We also need to test to see if the deliverable software and system will meet user and stakeholders needs and expectations in its operational environment (**validation**);
  - In every development cycle, a part of testing is focused on verification testing and a part is focused on validation testing.

# 1. FUNDAMENTALS OF TESTING

## What is testing?

- The process consisting of 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 demonstrate that they are fit for purpose and to detect defects.
- Typical objectives:
  - To evaluate work products such as requirements, user stories, design and code;
  - To verify whether all specified requirements have been fulfilled;
  - To validate whether the test object is complete and works as expected;
  - To build confidence in the level of quality of the test object.

# 1. FUNDAMENTALS OF TESTING

## What is testing?

- Typical objectives:
  - To prevent defects through the identification of defects in requirements specifications that are removed before they cause defects in the design specifications and subsequently on code;
  - To find failures and defects;
  - To provide sufficient information to stakeholders to allow them to make informed decisions;
  - To reduce the level of risk of inadequate software quality;
  - To comply with contractual, legal or regulatory requirements or standards and/or to verify the test object's compliance with such requirements or standards.

# 1. FUNDAMENTALS OF TESTING

## What is testing?

- Testing is not debugging:
  - While dynamic testing often locates failures which are caused by defects, and static testing often locates defects themselves, testing do not fix defects;
  - It is during debugging, a development activity, that a member of the project finds, analyses and removes the defect, the underlying cause of the failure;
  - After debugging there is a further testing activity associated with the defect, which is called confirmation testing.

# 1. FUNDAMENTALS OF TESTING

- Motivation;
- What is testing?
- **Why is testing necessary?**
- Seven testing principles;
- Test process.

# 1. FUNDAMENTALS OF TESTING

## Why is testing necessary?

- Testing's contributions to success:
  - The use of appropriate test techniques, applied with the right level of test expertise at the appropriate test levels and points in the software development cycle, can be of significant help in identifying problems so that they can be fixed before the software or systems is released into use.

# 1. FUNDAMENTALS OF TESTING

## Why is testing necessary?

- Quality assurance and testing:
  - Quality assurance is actually on part of a largest concept, quality management, which refers to all activities that direct and control a organization with regard to quality in all aspects;
  - Quality assurance is associated with ensuring that a company's standards ways of performing various tasks are carried out correctly;
  - Quality control is concerned with the quality of products rather than processes, to ensure that they achieved the desire level of quality;
  - Testing is looking at work products, including software, so it is actually a quality control activity rather than a quality assurance activity, despite common usage.

# 1. FUNDAMENTALS OF TESTING

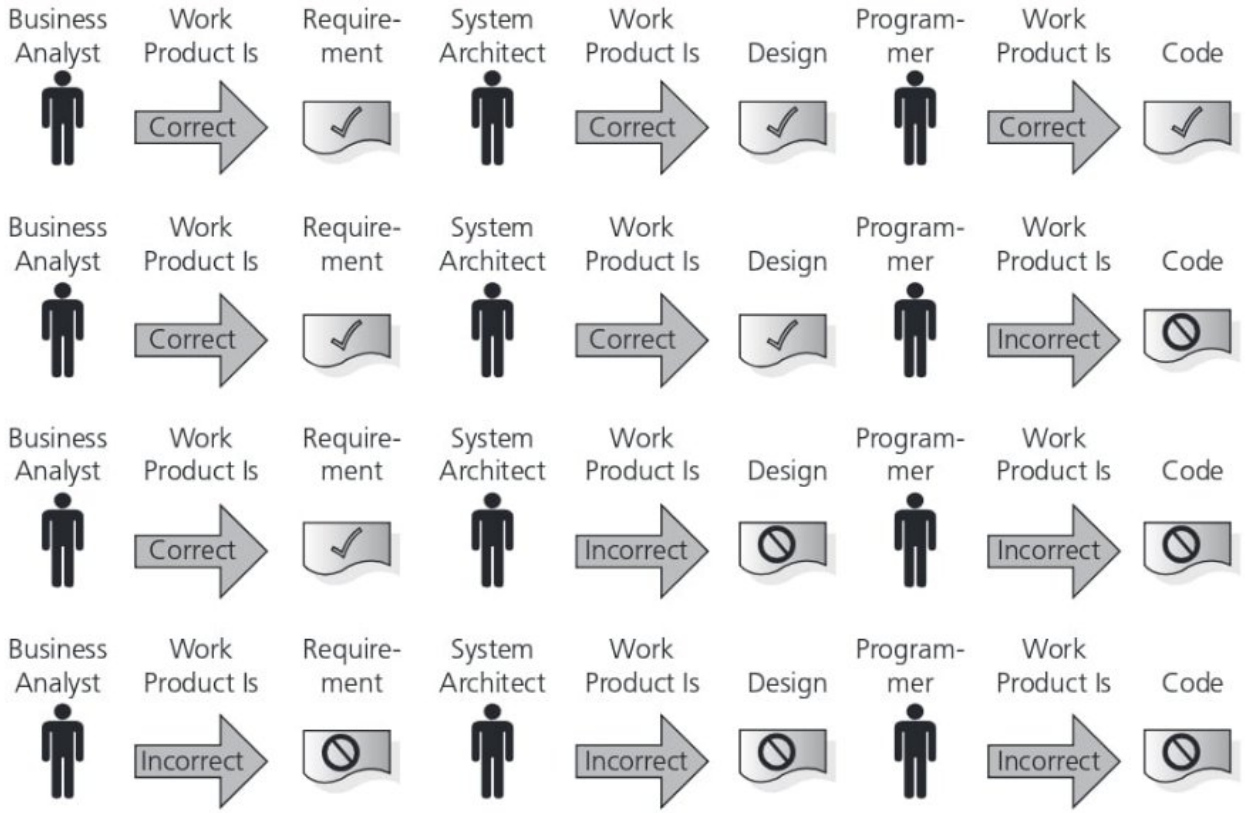
## Why is testing necessary?

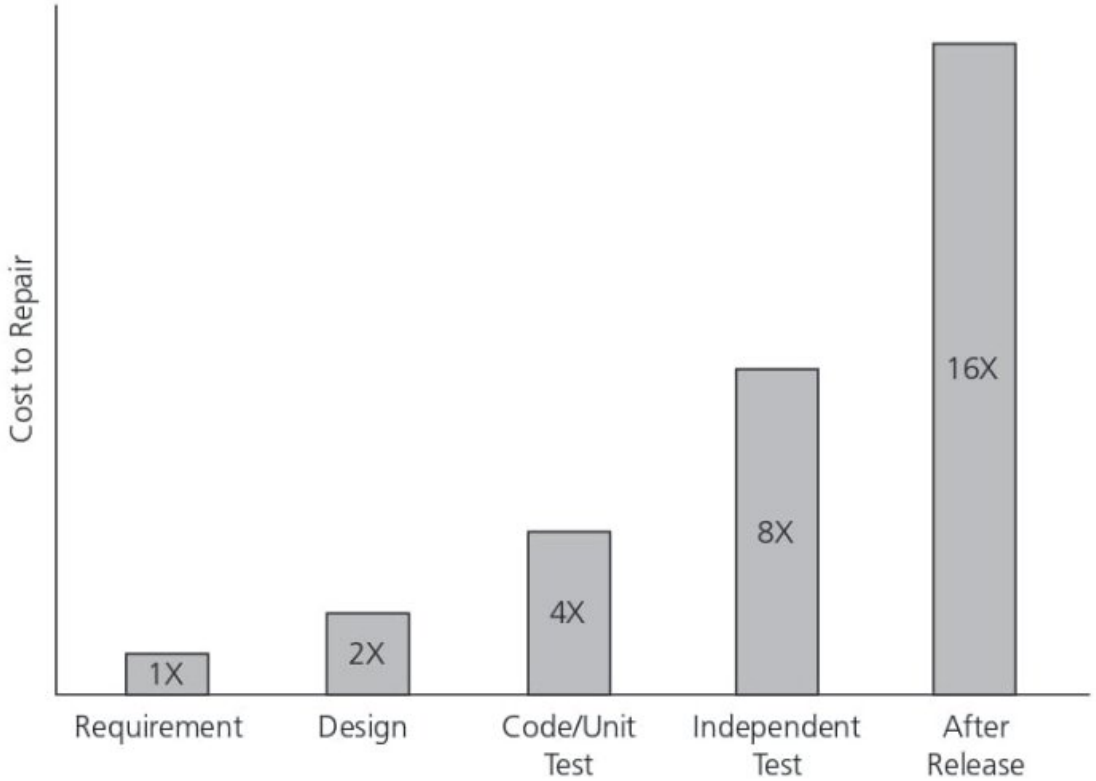
- Errors, defects and failures:
  - Errors (mistake): a human action that produces an incorrect result;
  - Defect (bug, fault): an imperfection or deficiency in a work product where it does not meet its requirements or specifications;
  - Failure: an event in which a component or system does not perform a required function within specifications limits.
  - Example: A developer makes an **error** such as forgetting about the possibility of inputting an excessively long string into a field. The developer thus puts a **defect** into the program, such as committing a check of input strings for length. When the program is executed this defect may result in unexpected behaviour, so the system exhibits a **failure**.

# 1. FUNDAMENTALS OF TESTING

## Why is testing necessary?

- Errors, defects and failures:



# 1. FUNDAMENTALS OF TESTING

## Why is testing necessary?

- Defects, root causes and effects:
  - A root cause is generally an organization issue, whereas a cause for a defect is an individual action;
  - Example: if a developers puts a “less than” instead a “greater than” symbol, this error may have been made through carelessness, but the carelessness may have been made worse because the intense time pressure to complete the module quickly. This can affects the user because it can produces unexpected behaviours in the software.

# 1. FUNDAMENTALS OF TESTING

- Motivation;
- What is testing?
- Why is testing necessary?
- **Seven testing principles;**
- Test process.

# 1. FUNDAMENTALS OF TESTING

## Seven testing principles

1. Testing shows the presence of defects, not their absence:
    - Testing can show that defects are present but cannot prove there are no defects;
    - Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, testing is not a proof of correctness.
2. Exhaustive testing is impossible:
    - Testing all combinations of inputs and preconditions is not feasible except for trivial cases;
    - Rather than attempting to test exhaustively, risk analysis, test techniques and priorities should be used to focus test efforts.

# 1. FUNDAMENTALS OF TESTING

## Seven testing principles

3. Early testing saves time and money:

- To find defects early, both static and dynamic test activities should start as early as possible;
- Testing early in the software development life cycle helps reduces or eliminate costly changes.

4. Defects cluster together:

- A small number of modules usually contains most of the defects discovered during pre-release testing, or they are responsible for most of the operational failures;
- Predicted defect clusters, and the actual observed defects clusters in test or operation, are an important input into a risk analysis used to focus the test effort.

# 1. FUNDAMENTALS OF TESTING

## Seven testing principles

5. Beware of the pesticide paradox:
    - It the same tests are repeated over and over again, eventually these tests no longer find any new defects;
    - To detect new defects, existing tests and test data are changed and new tests need to be written.
6. Testing is context dependent:
    - Testing is done differently in different contexts;
    - For example safety-critical software is tested differently from an e-commerce mobile app.
7. Absence-of-errors is a fallacy

# 1. FUNDAMENTALS OF TESTING

## Seven testing principles

7. Absence-of-errors is a fallacy:

- Some organizations expect that testers can run all possible test and find all possible defects, but principals 2 and 1 respectively tell us that is impossible;
- It is a fallacy to expect that just finding and fixing a large number of defects will ensure the success of a system.

# 1. FUNDAMENTALS OF TESTING

- Motivation;
- What is testing?
- Why is testing necessary?
- Seven testing principles;
- **Test process.**

# 1. FUNDAMENTALS OF TESTING

## Test process

- Each organization needs to adapt their test process depending on their context;
- Factors that influence the test process:
  - Software development life cycle and project methodologies;
  - Test levels and test types being considered;
  - Product and project risks;
  - Business domain;
  - Operational constrains (budget, resources, timescales, complexity, contractual and regulatory requirements);
  - Organizational policies and practices;
  - Required internal and external standards.

# 1. FUNDAMENTALS OF TESTING

## Test process

- **Test activities and tasks:**
  - Test planning;
  - Test monitoring and control;
  - Test analysis;
  - Test design;
  - Test implementation;
  - Test execution;
  - Test completion.

# 1. FUNDAMENTALS OF TESTING

## Test process

- **Test activities and tasks:**
  - **Test planning:** Involves defining the objectives of testing and the approach for meeting those objectives within project constrains and contexts. This includes deciding on suitable test techniques to use, deciding what tasks need to be done, formulating a test schedule and other things.
  - **Test monitoring and control:** In test monitoring, we continuously compare actual process against the plan, check on the progress of test activities and report the test status and any necessary deviations from the plan. In test control, we take whatever actions are necessary to meet the mission and objectives of the project, and/or adjust the plan.

# 1. FUNDAMENTALS OF TESTING

## Test process

- **Test activities and tasks:**
  - **Test analysis:** Involves the analysis of the test basis to identify testable features and define associated test conditions. Text analysis determines “what to test”, including measurable coverage criteria. The test basis can include requirements, user stories, design specifications, risk analysis reports, system design and architecture, interface and user expectations.
  - **Test design:** Addresses “how to test”, that is, what specific inputs and data are needed in order to exercise the software for a particular test condition. Test conditions are elaborated (at a high level) in test cases, sets of test cases and other testware.

# 1. FUNDAMENTALS OF TESTING

## Test process

- **Test activities and tasks:**
  - **Test implementation:** Specification of the test procedures (or test scripts). This involves combining the test cases in a particular order, as well as including any other information needed for test execution. Also involves setting up the test environment and anything else that needs to be done to prepare for test execution, such as creating testware.
  - **Test execution:** The test suites that have been assembled in test implementation are run, according to the test execution schedule.
  - **Test completion:** Involves the collection of data from completed test activities to consolidate experience, testware and any other relevant information.

# 1. FUNDAMENTALS OF TESTING

## Test process - Test work products

- Work products is the generic name given to any form of documentation, informal communication or artefact that is used in testing.
- **Test planning work products:** Test plans typically includes information about the test basis, entry and exit criteria (definition of ready and done) for the testing within their scope. It does **not** contain detail of test conditions, test cases or other aspects of testing.
- **Test monitoring and control work products:** Includes different types of test reports. Test progress reports are produced on an ongoing and/or regular basis to keep stakeholders updated about progress. Any test report needs to include details relevant to its intended audience, the date of the report and the time period covered. Test report may include test execution results once those are available.

# 1. FUNDAMENTALS OF TESTING

## Test process - Test work products

- **Test analysis work products:** Includes mainly test conditions, as this is the output of the test analysis activity. Each test condition is ideally traceable to the test basis (an vice versa). Defect reports about defects found in the test basis as a result of test analysis can also be considered a work product.
- **Test design work products:** The main work products resulting from test design are test cases and sets of test cases that exercise the test conditions identified in test analysis.
- **Test implementation work products:** Includes test procedures and the sequencing of those procedures, test suits, and a test execution schedule.

# 1. FUNDAMENTALS OF TESTING

## Test process - Test work products

- **Test execution work products:** Includes the documentation of the status of individual test cases or test procedures, defects reports, and documentation about which test item(s), test object(s), test tools and testware were involved during the test.
- **Test completion work products:** Includes the test summary reports, action items for improvement of subsequent projects or iterations, change requests or product backlog items, and finalized testware.

# 1. FUNDAMENTALS OF TESTING

## Test process - Traceability between the test basis and test work products

- Traceability is the degree in which a relationship can be established between two or more work products;
- We can trace a given requirement through test conditions and test cases to the test execution results but we can also trace the test execution results back to test cases, test cases back to test conditions, and test conditions back to requirements;
- The cross-linking gives many benefits to the test process.

# TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

- **Software development life cycle models;**
- Test levels;
- Test types.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models

- The life cycle model that is adopted for a project will have a big impact on the testing that is carried out;
- Test activities are highly related to software development activities;
- It will define the what, where, and when of our planned testing, influence regression testing and largely determine which test techniques to use;
- The way testing is organized must fit the development life cycle or it will fail to deliver its benefit.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

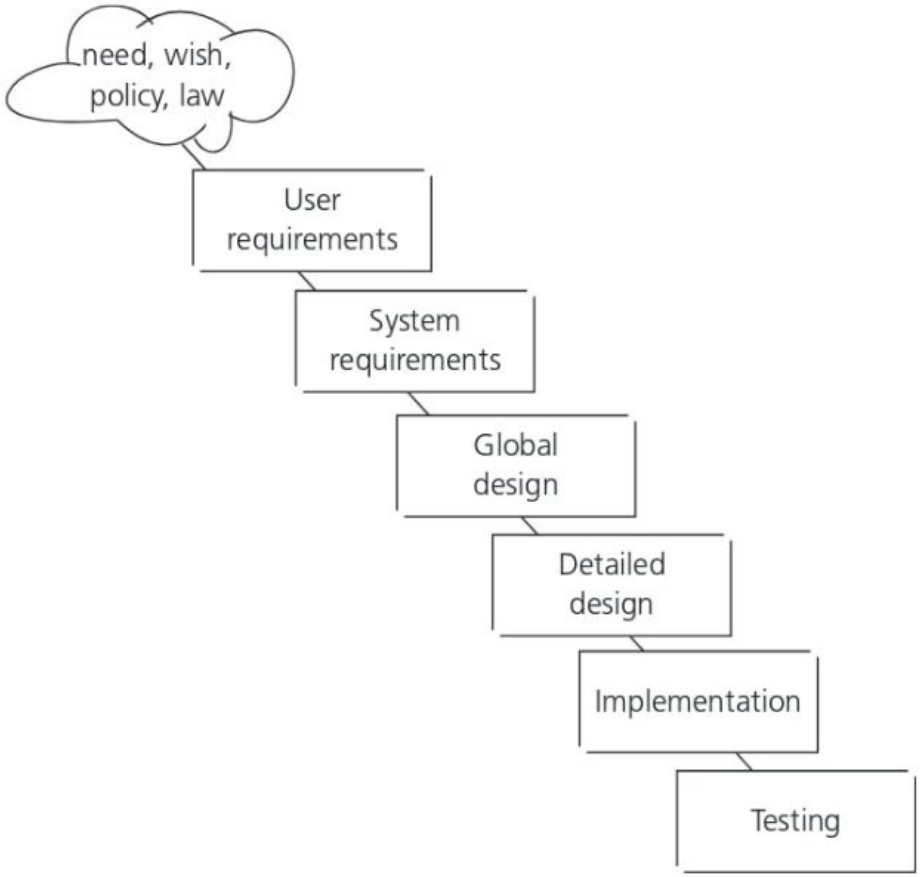
## Software development life cycle models

- Whichever life cycle model is being used, there are several characteristics of good testing:
  - For every development activity there is a corresponding test activity;
  - Each test level has test objectives specific to that level;
  - The analysis and design of tests for a given level should begin during the corresponding software development activity;
  - Testers should participate in discussions to help define and refine requirements and design.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models – Sequential models

- A sequential model is on where the development activities happen in a prescribed sequence – at least that is the idea;
- The models assume a linear sequential flow of activities;
- The next phase is only supposed to start when the previous phase is complete;
- The waterfall model was one of the earliest sequential models to be designed.



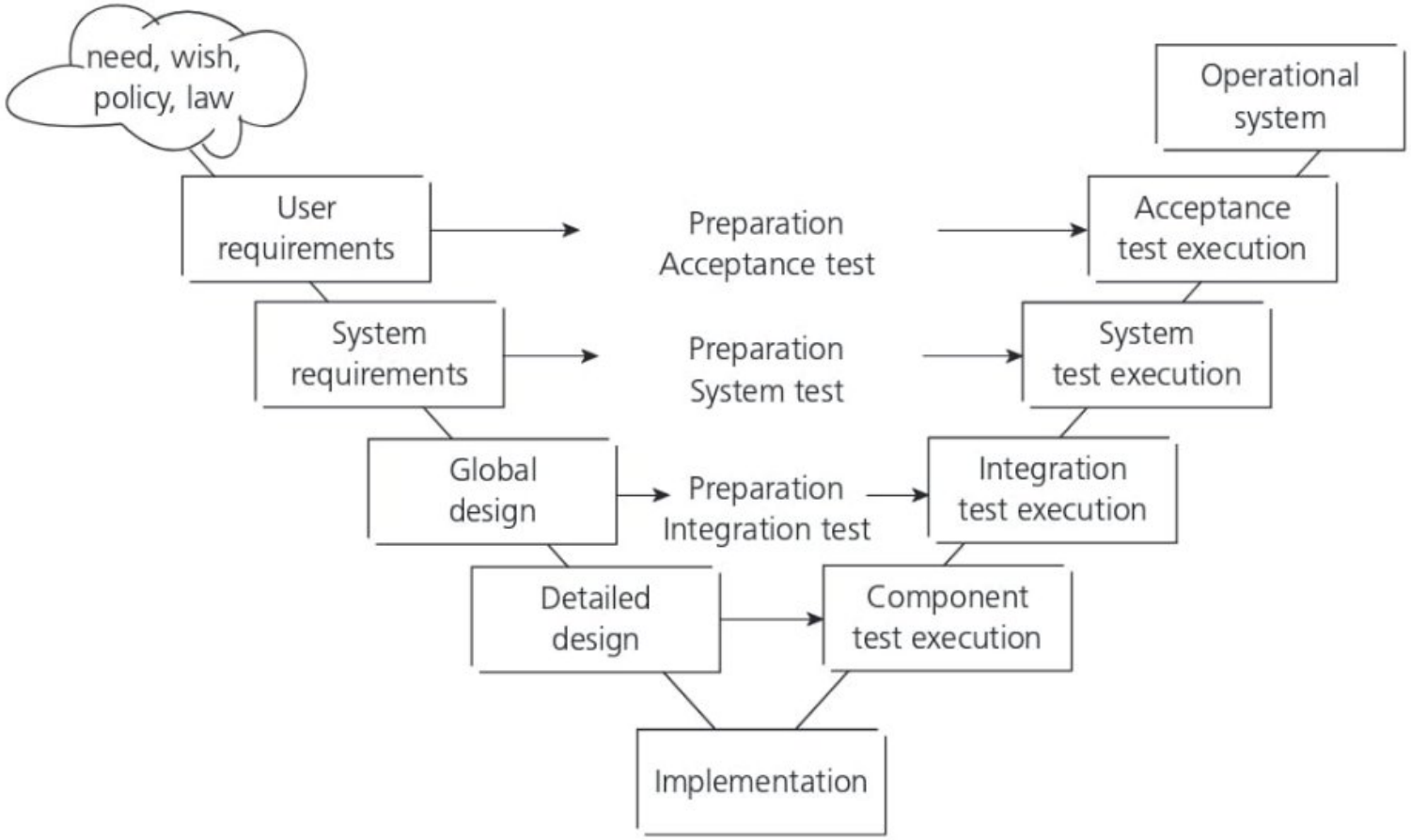
# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models – Sequential models

- In this models, defects were being found to late in the life cycle as testing was not involved until de end of the project;
- Testing also add lead time due to its late involvement;
- V-models provides guidance on how testing begins as early as possible in the life cycle;
- It also shows that testing is not only an execution-based activity;
- There are a variety of activities that need to be performed before the end of the coding phase;
- These activities should be carried out in parallel with development activities, and testers need to work with developers and analysts in order to perform them.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models – Sequential models



# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models – Iterative and incremental models

- In incremental models developers complete one piece at a time (scheduling or staging strategy) and each increment may be delivered to the customer;
- In iterative models developers start with a rough product and refine it, iteratively (rework strategy);
- In terms of developing software, a purely iterative model does not produce a working system until the final iteration;
- The incremental approach produces working versions of parts of the system early on and each of these can be released to the customer;
- In both iterative and incremental models, the features to be implemented are grouped together (for example according to business priority or risk).

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Software development life cycle models – Iterative and incremental models

- During project initiation, high-level test planning and test analysis occurs in parallel with the project planning and business/requirements analysis;
- Any detailed test planning, test analysis, test design and implementation occurs at the beginning of each iteration;
- Test execution often involves overlapping test levels;
- Each test level begins as early as possible and may continue after subsequent, higher test level have started;
- Many of these tasks will be performed but their timing and extent may vary.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

### Software development life cycle models – Iterative and incremental models

- Defects that are found in the current iteration/increment are dealt with the usual way;
- The defects that are found either by regression testing of previously parts, or discovered by accident need to be logged and dealt with, but because they are outside the scope of the current iteration/increment they can sometimes fall between the cracks and forgotten, neglected or argued about;
- There is a danger that the testing may be less thorough in incremental and iterative life cycles, particularly regression testing of previously developed parts;
- There is also a danger that the testing is less formal, because we are dealing with smaller parts of the system and formality of the testing may seem like overkill for such a small thing.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

- Software development life cycle models;
- **Test levels;**
- Test types.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

### Test Levels – Component Testing

- Also known as unit or modular testing, searches for defects in and verifies the functioning of, software items (modules, programs, objects, classes, functions, etc...) that are separately testable. Are typically based on the requirements and detailed design specifications applicable to the component under test, as well as the code itself.
- **Objectives:**
  - Reducing risk;
  - Verifying whether or not functional and non-functional behaviours of the component are as they should be;
  - Build confidence in the quality of the component;
  - Find defects in the component and prevent defects from escaping to later testing.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Component Testing

- **Test basis:**
  - Detailed design;
  - Code;
  - Data model;
  - Components specifications (if available).

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Component Testing

- **Test objects:**
  - Components themselves, units or modules;
  - Code and data structures;
  - Classes;
  - Database models.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Component Testing

- **Typical defects and failures:**
  - Incorrect functionality;
  - Data flow problems;
  - Incorrect code or logic.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Component Testing

- **Specific approaches and responsibilities:**
  - Typically, component testing occurs with access to the code being tested and with the support of the development environment, such as unit test framework or debugging tool;
  - In practice it usually involves the developer who wrote the code;
  - Defects are typically fixed as soon as they where found, without formally recording them in the defect management tool;
  - Although if such defects were recorded, this can provide useful information for the root cause analysis.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

### Test Levels – Integration Testing

- Tests interfaces between components and interactions of different parts of a system such as an operating system, file system and hardware or interfaces between systems. Are typically based on the software and system design, system architecture, and the work-flow.
- **Objectives:**
  - Reduce risk;
  - Verify whether or not functional and non-functional behaviours of the interfaces are as they should be, as designed and specified;
  - Building confidence in the quality of the interfaces;
  - Finding defects in the interfaces themselves or in the components or systems being tested together and prevent defects from escaping to later testing.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Integration Testing

- **Test basis:**
  - Software and system design;
  - Sequence diagrams;
  - Interface and communication protocol specifications;
  - Use cases;
  - Architecture at component or system level;
  - Work-flows;
  - External interface definitions.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Integration Testing

- **Test objects:**
  - Subsystems;
  - Databases;
  - Infrastructures;
  - Interfaces;
  - APIs;
  - Microservices.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Integration Testing

- **Typical defects and failures:**
  - Incorrect data, missing data or incorrect data encoding;
  - Incorrect sequencing or timing of interface calls;
  - Interface mismatch;
  - Failures in communication between components;
  - Unhandled or improperly handled communication failures between components;
  - Incorrect assumption about the meaning, units or boundaries of the data being passed between components;
  - Inconsistent message structures between systems;
  - Inconsistent data, missing data or incorrect data encoding.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Integration Testing

- **Typical defects and failures:**
  - Interface mismatch;
  - Failures in communication between systems;
  - Unhandled or improperly handled communication failures between systems;
  - Incorrect assumptions about the meaning, units or boundaries of the data being passed between systems;
  - Failure to comply with mandatory security regulations.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

### Test Levels – Integration Testing

- **Specific approaches and responsibilities:**
  - The greater the scope of integration, the more difficult it becomes to isolate failures to a specific interface which may lead to an increased risk;
  - Big Bang approach: all the components or modules are integrated together at once and them tested as unit (could become to complex to handle);
  - Incremental approach: testing are done by integrating two or more modules that are logically related to each other and them tested for proper functioning of the application.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- Is concerned with the behaviour of the whole system/product as defined by the scope of a development project or product. It may includes tests based on risk analysis report, system, functional or software requirements specifications, business processes, use cases or other high-level descriptions of system behaviour, interactions with the operating system and system resources.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- **Objectives:**
  - Reducing risk;
  - Verifying whether or not functional and non-functional behaviours of the system are as they should be;
  - Validating that the system is complete and will work as it should and as expected;
  - Building confidence in the quality of the system as a whole;
  - Finding defects and preventing defects from escaping to later testing or to production.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- **Test basis:**
  - Software and system requirement specifications;
  - Risk analysis report;
  - Use cases;
  - Epics and user stories;
  - Models of system behaviour;
  - State diagrams;
  - System and user manuals.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- **Test objects:**
  - Applications;
  - Hardware/software systems;
  - Operating systems;
  - Systems under test;
  - System configuration and configuration data.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- **Typical defects and failures:**
  - Incorrect calculations;
  - Incorrect or unexpected system functional or non-functional behaviour;
  - Incorrect control and/or data flows within the system;
  - Failure to properly and completely carry out end-to-end functional tasks;
  - Failure of the system to work properly in the production environment(s);
  - Failure of the system to work as described in system and user manuals.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – System Testing

- **Specific approaches and responsibilities:**
  - System testing is most often the final test on behalf of development to verify that the system to be delivered meets the specification and validate that it meets expectations;
  - One of its purposes is to find as many defects as possible;
  - System testing should investigate end-to-end behaviour of both functional and non-functional aspects of the system such as logging on, accessing data, placing order, logging off, (functional), but also performance, security and reliability (non-functional);
  - System testing requires a controlled test environment with regard to, among other things, control of the software versions, testware and the test data. Is frequently carried out by independent testers, internal test team or external testing specialists.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- Acceptance tests typically produce information to assess the system's readiness for release or deployment to end-users or customers. Although defects are found at this level, that is not the main aim of acceptance testing. The focus is on validation, the use of the system for real;
- **User Acceptance Testing (UAT):** it is acceptance testing done by (or on behalf of) end-users.
- **Operational Acceptance Testing (OAT):** focuses on operations and may be performed by systems administrators such as installing, backups, disaster recovery, etc...
- **Contractual and Regulatory Acceptance Testing:** is focused on whether or not the system meets contractual or regulatory criteria.
- **Alpha and Beta Testing:** the main goal is looking for feedback and defects from real users and customers.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Objectives:**
  - Establishing confidence in the quality of the system as a whole;
  - Validating that the system is complete and will work as expected;
  - Verifying that functional and non-functional behaviours of the system are as specified.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Test basis:**
  - Business processes;
  - User or business requirements;
  - Regulations, legal contracts and standards;
  - Use cases;
  - System requirements;
  - System or user documentation;
  - Installation procedures;
  - Risk analysis report.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Test basis:**
  - Backup and restore/recovery procedures;
  - Disaster recovery procedures;
  - Non-functional requirements;
  - Operations documentation;
  - Deployment and installation instructions;
  - Performance targets;
  - Database packages;
  - Security standards and regulations.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Test objects:**
  - System under test;
  - System configuration and configuration data;
  - Business processes for a fully integrated system;
  - Recovery systems;
  - Operational and maintenance processes;
  - Forms;
  - Reports;
  - Existing and converted production data.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Typical defects and failures:**
  - System work-flows do not meet business or users requirements;
  - Business rules are not implemented correctly;
  - System does not satisfy contractual or regulatory requirements;
  - Non-functional failures such as security vulnerabilities, inadequate performance efficiency under high load, or improper operation on a supported platform.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Levels – Acceptance Testing

- **Specific approaches and responsibilities:**
  - The acceptance test should answer questions such as:
    - Can the system be released?
    - What, if any, are the outstanding (business) risks?
    - Has development met their obligations?
  - Acceptance testing is most often the responsibility of the user or customer, although other stakeholders may be involved as well;
  - The execution of the acceptance test requires a test environment that is, for most aspects, representative of the production environment.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

- Software development life cycle models;
- Test levels;
- **Test types.**

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Types – Functional Testing

- Testing conducted to evaluate the compliance of a component or system with functional requirements;
- Functional testing considers the specified behaviour and is often also referred to as black-box testing (specification-based testing). This is not entirely true, since black-box testing also includes non-functional testing;
- Functional testing can also be done focusing on suitability, interoperability testing, security, accuracy and compliance;
- Testing of functionality could be done from different perspectives, the two main are ones being requirements-based or business-process-bases.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Types – Non-functional Testing

- Testing conducted to evaluate the compliance of a component or a system with non-functional requirements;
- Includes, but is not limited to, performance testing, reliability testing, load testing, stress testing, usability testing, maintainability testing, reliability testing, portability testing, and security testing;
- It is the testing of how well the system works.

## 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

### Test Types – White-box Testing

- Also known as clear-box testing, code-base testing, logic-coverage testing, glass-box testing, logic-driven testing, structural testing, structure-based testing. Testing based on an analysis of the internal structure-based of the component or system;
- Looks at the internal structure or implementations of the system or component. Includes the structure of a system (system architecture), code itself, control flows, business processes and data flows;
- Is most often used as a way of measuring the thoroughness of testing through the coverage of a set of structural elements or coverage items.

# 2. TESTING THROUGHOUT THE SOFTWARE DEVELOPMENT LIFE CYCLE

## Test Types – Change-related Testing

- **Confirmation testing (re-testing):** dynamic testing conducted after fixing defects with the objective to confirm that failures caused by those defects do not occur anymore. It is important to ensure that steps leading up to the failure are carried out in exactly the same way as described in the defect report.
- **Regression testing:** testing of a previously tested component or system following modification to ensure that defects have not been introduced or have been uncovered in unchanged areas of the software as a result of the changes made. Involves executing test cases that have been executed before.

# TEST TECHNIQUES

## 3. TEST TECHNIQUES

- **Black-box test techniques;**
- White-box test techniques;
- Experienced-based test techniques.

# 3. TEST TECHNIQUES

## Black-box test techniques

- They are called black-box techniques because they view the software as a black box with inputs and outputs, but they have now knowledge of how the component or system works inside the box;
- In its essence the tester is concentrating on what the software does, not how it does it;
- Common characteristics of black-box test techniques include the following:
  - Test conditions, test cases and test data that are derived from a test basis that may include software requirements, specifications, use cases, and user stories;
  - Test cases may be used to detect gaps between the requirements and the implementation of the requirements, as well as deviations from the requirements;
  - Coverage is measured based on the items tested in the test basis and techniques applied.

# 3. TEST TECHNIQUES

## Black-box test techniques

- Black-box test techniques are appropriate at all levels of testing where a specification or other test basis exists;
- Can be used to test both functional or non-functional specifications. Functional testing is concerned with what the system does, its features and functions and non-functional testing is concerned with examining how well the system does something.

## 3. TEST TECHNIQUES

- Black-box test techniques;
- **White-box test techniques;**
- Experienced-based test techniques.

# 3. TEST TECHNIQUES

## White-box test techniques

- White-box test techniques use the internal structure of the software to derive test cases;
- They are commonly called white-box or glass-box techniques since they require knowledge on how the software is implemented, that is, how it works;
- Common characteristics of white-box test techniques include the following:
  - Test conditions, test cases and test data are derived from a test basis that may include code, software architecture, detailed design or other source of information regarding the structure of the software;
  - Coverage is measured based on the items tested within a selected structure;
  - Determine the path through the software that either was taken or that you want to be taken, and this is determined by specific inputs to the software.

# 3. TEST TECHNIQUES

## White-box test techniques

- Can be used at all levels of testing;
- Developers use white-box techniques in component testing and component integration testing;
- Are also used in system and acceptance testing, but the structures are different.

## 3. TEST TECHNIQUES

- Black-box test techniques;
- White-box test techniques;
- **Experienced-based test techniques.**

# 3. TEST TECHNIQUES

## Experience-based test techniques

- In experience-based test techniques, people's knowledge, skills and background are a prime contributor to the test conditions, test cases and test data;
- The experience of both technical and business people is important, as they bring different perspectives to the test analysis and design process;
- Due to previous experience with similar systems, they may have insights into what could go wrong, which is very useful for testing;
- They are based on human knowledge and experience;
- Test cases are therefore derived in a less systematic way, but may be more effective;
- Are used to complement black-box and white-box techniques and are also used when there is no specification, or if the specification is inadequate or out-of-data.

# TEST MANAGEMENT

## 4. TEST MANAGEMENT

- **Test organization;**
- Test planning and estimation;
- Test monitoring and control;
- Configuration management;
- Risks and testing;
- Defect management.

# 4. TEST MANAGEMENT

## Test organization

- Testing tasks may be done by people with a specific testing role, but testers are definitely not the only people who do testing. Developers, business analysts, users and customers also do testing tasks for different reasons and at different times;
- Many organizations, especially in safety-critical areas, have separate teams of independent testers because without cognitive bias this independent testers can be much more effective finding defects;
- An independent tester can often see more, different defects than a tester working within a development team – or a tester who is by profession a developer;
- Independent teams are not risk-free. It is possible for the testers and the test team to become isolated from the developers, the designers, and the project team itself.

# 4. TEST MANAGEMENT

## Test organization

- The use of independent teams or internal teams should be carefully managed by organizations;
- Common testing roles:
  - **Test manager:** the person responsible for project management of testing activities and resources, and evaluation of a test object. The individual who directs, controls, administers, plans and regulates the evaluation of a test object;
  - **Tester:** a skilled professional who is involved in the testing of a component or system.

## 4. TEST MANAGEMENT

- Test organization;
- **Test planning and estimation;**
- Test monitoring and control;
- Configuration management;
- Risks and testing;
- Defect management.

# 4. TEST MANAGEMENT

## Test planning and estimation

- **Test plan:** documentation describing the test objectives to be achieved and the means and the schedule for achieving them, organized to coordinate testing activities;
- **Test planning:** is the activity of establishing or updating a test plan;
- **Test strategy:** documentation that expresses the generic requiremnts for testing one or more projects run within an organization, providing detail on how testing is to be performed, and is aligned with the test policy;
- **Test approach:** the implementation of the test strategy for a specific project;
- **Entry criteria:** the set of conditions for officially starting a defined task (e.g. Availability of testable requirements, test items, test environment, etc...).

# 4. TEST MANAGEMENT

## Test planning and estimation

- **Exit criteria:** set of conditions for officially completing a defined task (number of tests, coverage, number of defects, schedule, etc...);
- **Test execution schedule:** a schedule for the execution of the test suites based on the priorities pf the tests, technical or logical dependencies of tests of tests suites, and the type of tests;
- **Test estimation:** the calculated approximation of a result related to various aspects of testing (e.g. effort spent, completion date, costs, number of test cases, etc...), which is usable even if input data may be incomplete, uncertain or noisy.

## 4. TEST MANAGEMENT

- Test organization;
- Test planning and estimation;
- **Test monitoring and control;**
- Configuration management;
- Risks and testing;
- Defect management.

# 4. TEST MANAGEMENT

## Test monitoring and control

- **Test monitoring:** a test management activity that involves checking the status of testing activities, identifying any variances from the planned or expected status and reporting status to stakeholders;
- **Test control:** a test management task that deals with developing and applying a set of corrective actions to get a test project on track when monitoring shows a deviation from what was planned;
- **Common metrics:** the percentage of planned work done in test case preparation; percentage of planned work done in test environment preparation; number o test cases run and not run; number o test cases passed or failed; number of test conditions passed or failed; defects density; defects found and fixes; test coverage, etc...;

# 4. TEST MANAGEMENT

## Test monitoring and control

| | Unresolved defects | | Test cases to be run | | |
|---|---|---|---|---|---|
| **Product risk areas** | **Number** | **%** | **Planned** | **Actual** | **%** |
| *Performance, load, reliability* | 304 | 28 | 3,843 | 1,512 | 39 |
| *Robustness, operations, security* | 234 | 21 | 1,032 | 432 | 42 |
| *Functionality, data, dates* | 224 | 20 | 4,744 | 2,043 | 43 |
| *Use cases, user interfaces, localization* | 160 | 15 | 498 | 318 | 64 |
| *Interfaces* | 93 | 8 | 193 | 153 | 79 |
| *Compatibility* | 71 | 6 | 1,787 | 939 | 53 |
| *Other* | 21 | 2 | 760 | 306 | 40 |
| | *1,107* | *100* | *12,857* | *5,703* | *44* |

- **Test progress report:** a test report produced at regular intervals about the progress of test activities against a baseline, risks and alternatives requiring a decision;
- **Test summary report:** a test report that provides an evaluation of the corresponding test items against exist criteria.

## 4. TEST MANAGEMENT

- Test organization;
- Test planning and estimation;
- Test monitoring and control;
- **Configuration management;**
- Risks and testing;
- Defect management.

# 4. TEST MANAGEMENT

## Configuration management

- A discipline applying technical and administrative direction and surveillance to identify and document the functional an physical characteristics of a configuration item, control changes to those characteristics record and report change processing and implementation status, and verify compliance with specified requirements.

## 4. TEST MANAGEMENT

- Test organization;
- Test planning and estimation;
- Test monitoring and control;
- Configuration management;
- **Risks and testing;**
- Defect management.

# 4. TEST MANAGEMENT

## Risks and testing

- **Risk:** a factor that could result in future negatives consequences;
- **Risk level:** the qualitative or quantitative measure of a risk defined by impact and likelihood;
- **Product risk:** a risk impacting the quality of a product:
  - Software might not perform its intended functions according to the specification;
  - A particular computation may be performed incorrectly in some circumstances;
  - A loop control structures may be coded incorrectly;
  - User experience feedback might not meet product expectations.
- **Project risk:** a risk that impacts project success.

# 4. TEST MANAGEMENT

## Risks and testing

- **Risk management:** dealing with risks within an organization is know as risk management and testing is one way of managing aspects of risk. For any risk, product or project, there are four typical options:
  - Mitigate: take steps in advance to reduce the likelihood (and possibly the impact) of a risk;
  - Contingency: have a plan in place to reduce the impact of the risk;
  - Transfer: transfer the risk to other entities;
  - Ignore: do nothing about the risk.
- **Risk-based testing:** testing in which the management, selection, prioritization and use of testing activities and resources are based on corresponding risks types and risk levels.

# 4. TEST MANAGEMENT

## Risks and testing

| Product risk | Likelihood | Impact | Risk priority number | Mitigation |
|---|---|---|---|---|
| *Risk category 1* | | | | |
| *Risk 1* | | | | |
| *Risk 2* | | | | |
| *Risk n* | | | | |

## 4. TEST MANAGEMENT

- Test organization;
- Test planning and estimation;
- Test monitoring and control;
- Configuration management;
- Risks and testing;
- **Defect management.**

# 4. TEST MANAGEMENT

## Defect management

- The process of recognizing and recording defects, classifying them, investigating them, taking action to resolve them and disposing of them when resolved.