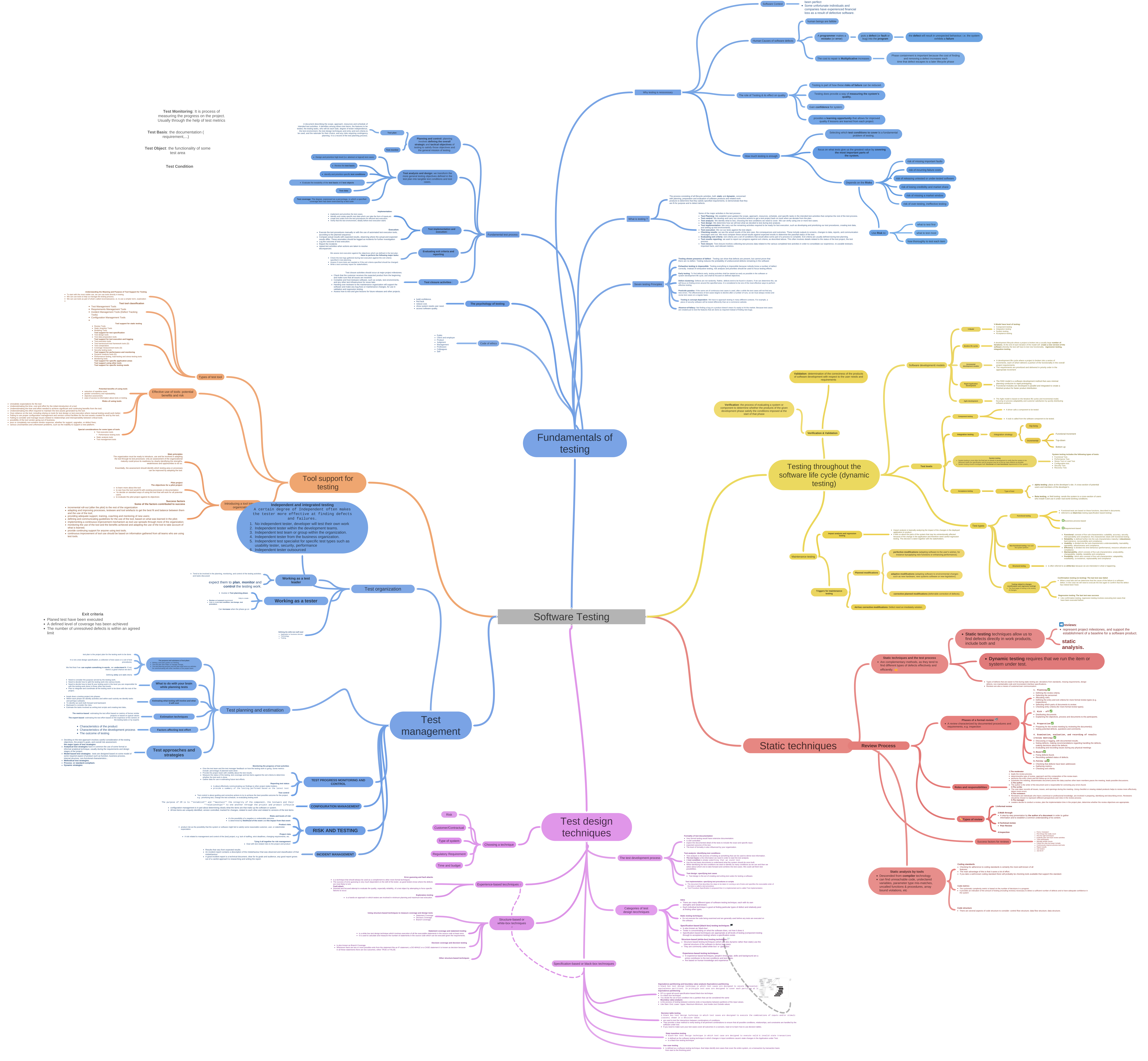
coggle
made for free at coggle.it



Software Testing

Test Monitoring: It is process of measuring the progress on the project. Usually through the help of test metrics

Test Basis: the documentation (requirement,...)

Test Object: the functionality of some test area

Test Condition

Fundamentals of testing

Why testing is nesscessary

Software Context
- Software system is very necessary in life
- However, the technological triumphhas not been perfect
- Some unfortunate individuals and companies have experienced financial loss as a result of defective software.

Human Causes of software defects
- human beings are fallible
- A **programmer** makes a **mistake** (or **error**) → puts a **defect** (or **fault** or bug) into the **program** → the **defect** will result in unexpected behaviour; i.e. the system exhibits a **failure**
- The cost to repair is **Multiplicative** increases → Phase containment is important because the cost of finding and removing a defect increases each time that defect escapes to a later lifecycle phase

The role of Testing & its effect on quality
- Testing is part of how these **risks of failure** can be reduced.
- Testing does provide a way of **measuring the system's quality.**
- Gain **confidence** for system
- provides a **learning opportunity** that allows for improved quality if lessons are learned from each project.

How much testing is enough
- Selecting which **test conditions to cover** is a fundamental problem of testing.
- focus on what tests give us the greatest value by **covering the most important parts of the system.**
- Depends on the **Risks**
  - risk of missing important faults
  - risk of incurring failure costs
  - risk of releasing untested or under-tested software
  - risk of losing credibility and market share
  - risk of missing a market window
  - risk of over-testing, ineffective testing
  - Use **Risk** to:
    - what to test first
    - what to test most
    - how thoroughly to test each item

What is testing ?

The process consisting of all lifecycle 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.

Some of the major activities in the test process :
- **Test Planning**: We establish (and update) the scope, approach, resources, schedule, and specific tasks in the intended test activities that comprise the rest of the test process.
- **Test control**: We develop and carry out corrective actions to get a test project back on track when we deviate from the plan.
- **Test analysis**: We identify what to test, choosing the test conditions we need to cover. We can verify using one or more test cases.
- **Test design**: We determine how we will test what we decided to test during test analysis.
- **Test implementation**: We carry out the remaining activities required to be ready for test execution, such as developing and prioritizing our test procedures, creating test data, and setting up test environments.
- **Test execution**: We run our tests against the test object.
- **Checking results**: we see the actual results of the test case, the consequences and outcomes. These include outputs to screens, changes to data, reports, and communication messages sent out. We must compare these actual results against expected results to determine the pass/fail status of the test.
- **Evaluating exit criteria**: exit criteria are a set of conditions that would allow some part of a process to complete. Exit criteria are usually defined during test planning.
- **Test results reporting**: we want to report our progress against exit criteria, as described above. This often involves details related to the status of the test project, the test process.
- **Test closure**: Test closure involves collecting test process data related to the various completed test activities in order to consolidate our experience, re-useable testware, important facts, and relevant metrics.

Seven testing Principles

- **Testing shows presence of defect** : Testing can show that defects are present, but cannot prove that there are no defect. Testing reduces the probability of undiscovered defects remaining in the software
- **Exhautive testing is impossible**: Testing everything is impossible because nobody know a number of defect correctly. Instead of exhaustive testing, risk analysis and priorities should be used to focus testing efforts.
- **Early testing:** To find defects early, testing activities shall be started as early as possible in the software or system development life cycle, and shall be focused on defined objectives.
- **Defect clustering:** Defects are not randomly. Rather, defects tend to be found in clusters. If we can determine this, we will focus on finding errors around the specified area. It is considered to be one of the most effective ways to perform effective testing.
- **Pesticide paradox:** If the same set of continuous test cases is used, after a while the test cases will not find any new errors. The effectiveness of test cases begins to decline after a number of runs, so we must always review and revise test cases on a regular basis.
- **Testing is concept dependent:** We have to approach testing in many different contexts. For example, a piece of security software will be tested differently than an e-commerce website.
- **Absence-of-fallacy:** Not finding a bug on a product doesn't mean it's ready to hit the market. Because test cases are created just to test the features that are done as required instead of finding new bugs.

Fundamental test process

Planning and control: planning involves **defining the overall strategic** and **tactical objectives** of testing to satisfy those objectives and the general mission of testing.
- **Test plan**: A document describing the scope, approach, resources and schedule of intended test activities. It identifies among others test items, the features to be tested, the testing tasks, who will do each task, degree of tester independence, the test environment, the test design techniques and entry and exit criteria to be used, and the rationale for their choice, and any risks requiring contingency planning. It is a record of the test planning process.
- **Test monitor**

Test analysis and design: we transform the more general testing objectives defined in the test plan into tangible test conditions and test cases.
- Design and prioritize high level (i.e. abstract or logical) test cases
- Review the **test basis.**
- Identify and prioritize specific **test conditions**
- Evaluate the testability of the **test basis** and **test objects**
- **Test data**
- **Test coverage**: The degree, expressed as a percentage, to which a specified coverage item has been exercised by a test suite

Test implementation and execution:

Implementation:
- implement and prioritize the test cases.
- Identify and create specific test data which can take the form of inputs,etc
- create test suites from the test procedures for efficient test execution.
- Verify that the test environment, ideally before test execution starts

Execution:
- Execute the test procedures manually or with the use of automated test execution tools, according to the planned sequence.
- Compare actual results with expected results, observing where the actual and expected results differ. These anomalies should be logged as incidents for further investigation
- Log the outcome of test execution.
- Report the incidents
- repeat test activities when actions are taken to resolve discrepancies

Evaluating exit criteria and reporting

We assess test execution against the objectives which we defined in the test plan.
Have to perform the following major tasks:
- Check the test logs gathered during test execution against the exit criteria specified in test planning.
- Assess if more tests are needed or if the exit criteria specified should be changed.
- Write a test summary report for stakeholders.

Test closure activities :

Test closure activities should occur at major project milestones.
- Check that the customer receives the expected product from the beginning and make sure that all issues are resolved.
- Complete and host testware software, such as scripts, test environments, and any other test infrastructure for reuse later.
- Handing over testware to the maintenance organization will support the software and make any bug fixes or maintenance changes, for use in validation and regression testing.
- Assess how to test and give lessons for future releases and other projects.

The psychology of testing

- build confidence.
- find fault.
- reduce cost.
- show system needs user need.
- access software quality.

Code of ethics

- Public
- Client and employer
- Product
- Judgment
- Management
- Profession
- Colleagues
- Self

Testing throughout the software life cycle (dynamic testing)

Verification & Validation

Validation: determination of the correctness of the products of software development with respect to the user needs and requirements

Verification: the process of evaluating a system or component to determine whether the products of the given development phase satisfy the conditions imposed at the start of that phase

Software developmentt models

V-Model
V-Model have level of testing:
- Component testing
- Integration testing
- System testing
- Acceptance testing

Iterative life cycles
A development lifecycle where a project is broken into a usually large **number of iterations.** At the end of each iteration of the model will **create a new version of the software** whereby the test will have to test new functionality, **regression testing, integration testing.**

Incremental development models
- A development life cycle where a project is broken into a series of increments, each of which delivers a portion of the functionality in the overall project requirements.
- The requirements are prioritized and delivered in priority order in the appropriate increment

Rapid Application Development
- The RAD model is a software development method that uses minimal planning conducive to rapid prototyping.
- Functional modules are developed in parallel and integrated to create a finished product for faster product distribution.

Agile development
- The Agile model is based on the Iterative life cycles and Incremental model, focusing on process adaptability and customer satisfaction by quickly distributing software products.

Test levels

Component testing
- A driver calls a component to be tested
- A stub is called from the software component to be tested.

Integration testing → Integration strategy
- big-bang
- incremental
  - Functional Increment
  - Top-down
  - Bottom up

System testing
- System testing is most often the final test on behalf of development to verify that the system to be delivered meets the specification and its purpose may be to find as many defects as possible
- System testing should investigate both **functional** and **non-functional** requirements of the system.

System testing includes the following types of tests:
- Functional Test.
- Performance Test.
- Stress Test or Load Test
- Configuration test.
- Security Test.
- Recovery Test.

Acceptance testing → Type of test
- **alpha testing**: place at the developer's site. A cross-section of potential users and members of the developer's
- **Beta testing,** or field testing: sends the system to a cross-section of users who install it and use it under real-world working conditions.

Test types

Functional testing
- Functional tests are based on these functions, described in documents.
- referred to as **black-box** testing (specification based testing).
- ☒Bussiness-process-based
- ☒Requirement-based

Non-functional testing: how well the system perform
- **Functional**: consists of five sub-characteristics: suitability, accuracy, security, interoperability and compliance; this characteristic deals with functional testing.
- **Reliability**: is defined further into the sub-characteristics maturity (**robustness**), fault-tolerance, recoverability and compliance;
- **Usability**: is divided into the sub-characteristics understandability, learnability, operability, attractiveness and compliance.
- **Efficiency**: is divided into time behaviour (performance), resource utilization and compliance.
- **Maintainability**, which consists of five sub-characteristics: analyzability, changeability, stability, testability and compliance.
- **Portability**: which also consists of five sub-characteristics: adaptability, installability, co-existence, replaceability and compliance

Structural testing
- is often referred to as **white-box** because we are interested in what is happening.

Testing related to changes (confirmation and regression testing)
- The final target of testing is the testing of changes.

Confirmation testing (re-testing) The last test was failed
- When a test fails and we determine that the cause of the failure is a software defect. In this case we will need to execute the test again to confirm that the defect has indeed been fixed.

Regression testing The last test was success
- Like confirmation testing, regression testing involves executing test cases that have been executed before.

Maintenance testing

Impact analysis and regression testing
- Impact analysis is basically analyzing the impact of the changes in the deployed application or product.
- It tells us about the parts of the system that may be unintentionally affected because of the change in the application and therefore need careful regression testing. This decision is taken together with the stakeholders.

Triggers for maintenance testing
- Planned modifications
  - **perfective modifications** (adapting software to the user's wishes, for instance bysupplying new functions or enhancing performance);
  - **adaptive modifications** (adapting software to environmental changes such as new hardware, new systems software or new legislation);
  - **corrective planned modifications** (deferrable correction of defects).
- **Ad-hoc corrective modifications**: Defect need an imediately solution.

Static techniques

Static techniques and the test process

- Are complementary methods, as they tend to find different types of defects effectively and efficiently. 🔒
- **Static testing** techniques allow us to find defects directly in work products, include both and
  - **reviews**:
    - represent project milestones, and support the establishment of a baseline for a software product.
  - **static analysis.**
- **Dynamic testing** requires that we run the item or system under test.
- Types of defects that are easier to find during static testing are: deviations from standards, missing requirements, design defects, non-maintainable code and inconsistent interface specifications.
- Reviews are also a means of customer/user communication.

Review Process

Phases of a formal review 🏁
- A review characterized by documented procedures and requirements, e.g. inspection

`1. Planning` ✅
- Defining the review criteria.
- Selecting the personnel.
- Allocating roles.
- Defining the entry and exit criteria for more formal review types (e.g. inspections).
- Selecting which parts of documents to review.
- Checking entry criteria (for more formal review types).

`2. Kick - off` ✅
- Distributing documents.
- Explaining the objectives, process and documents to the participants.

`3. Preparation` ✅
- Preparing for the review meeting by reviewing the document(s).
- Noting potential defects, questions and comments.

`4. Examination, evaluation, and recording of results (review meeting)` ✅
- Discussing or logging, with documented results.
- Noting defects, making recommendations regarding handling the defects, making decisions about the defects.
- Evaluating and recording issues during any physical meetings

`5.Rework` ✅
- Fixing defects found.
- Recording updated status of defects.

`6.Follow up` ✅
- Checking that defects have been addressed.
- Gathering metrics.
- Checking exit criteria.

Roles and responsibilities

1.The moderator
- leads the review process.
- determinesthe type of review, approach and the composition of the review team.
- performs the entry check and the follow-up on the rework.
- schedules the meeting, disseminates document,stores the data,coaches other team members,paces the meeting, leads possible discussions.

2.The author
- The author is the writer of the document and is responsible for correcting any errors found.

3.The scribe
- The note taker records all issues, issues, and openings during the meeting. Using checklist or viewing related products helps to review more effectively, detect more errors.

4.The reviewers
- Reviewers are individuals who have a technical or professional knowledge, are involved in preparing, identifying and describing errors. Reviewers should be chosen to represent different perspectives and roles in the review process.

5.The manager
- Leaders decide to conduct a review, plan the implementation time in the project plan, determine whether the review objectives are appropriate.

✏️**Types of review**
- **1.Informal review**
- **2.Walk through**
  - A step-by-step presentation by **the author of a document** in order to gather information and to establish a common understanding of its content.
- **3.Technical review**
  - **Peer Review**
- **4.Inspection**

Success factors for reviews
- Find a 'champion'
- Pick things that really count
- Pick the right techniques
- Explicitly plan and track review activities
- Train participants
- Manage people issues
- Follow the rules but keep it simple
- Continuously improve process and tools
- Report results
- Use testers
- Just do it!

Static analysis by tools

- Descended from **compiler** technology
- can find unreachable code, undeclared variables, parameter type mis-matches, uncalled functions & procedures, array bound violations, etc

Coding standards
- Checking for adherence to coding standards is certainly the most well-known of all features.
- The main advantage of this is that it saves a lot of effort.
- if you take a well-known coding standard there will probably be checking tools available that support this standard.

Code metrics
- The cyclomatic complexity metric is based on the number of decisions in a program.
- It provides an indication of the amount of testing (including reviews) necessary to detect a sufficient number of defects and to have adequate confidence in the system

Code structure
- There are several aspects of code structure to consider: control flow structure; data flow structure; data structure.

Test design techniques

Choosing a technique

- Risk
- Customer/Contractual
- Type of system
- Regulatory Requirement
- Time and budget

The test development process

Formality of test documentation
- Very formal testing would have extensive documentation.
- is well controlled.
- expect the documented detail of the tests to include the exact and specific input.
- expected outcome of the test.
- The level of formality is also influenced by your organization.

Test analysis: identifying test conditions
- Test analysis is the process of looking at something that can be used to derive test information.
- **The test basis** is the information we need in order to start the test analysis.
- A **test condition** is simply `something that we could test`
- Use the test basis documents to understand what the system should do once built.
- While identifying the test conditions we want to identify as many conditions as we can and then we select about which one to take forward and combine into test cases. We could call them test possibilities.

Test design: specifying test cases
- Test design is the act of creating and writing test suites for testing a software.

Test implementation: specifying test procedures or scripts
- The document that describes the steps to be taken in running a set of tests and specifies the executable order of the tests is called a test procedure.
- Test Procedure Specification is prepared then it is implemented and is called Test implementation.

Categories of test design tecchniques

Intro
- There are many different types of software testing technique, each with its own strengths and weaknesses.
- Each individual technique is good at finding particular types of defect and relatively poor at finding other types.

Static testing techniques
- Do not execute the code being examined and are generally used before any tests are executed on the software.

Specification-based (black-box) testing techniques 🏴
- Is also known as 'black-box'.
- Tester is concentrating on what the software does, not how it does it.
- Specification-based techniques are appropriate at all levels of testing (component testing through to acceptance testing) where a specification exists

Structure-based (white-box) testing techniques 🏳
- Structure-based testing techniques (which are also dynamic rather than static) use the internal structure of the software to derive test cases.
- They are commonly called white-box' or 'glass-box'.

Experience-based testing techniques
- In experience-based techniques, people's knowledge, skills and background are a prime contributor to the test conditions and test cases.
- Are based on human knowledge and experience

Experience-based tecchniques

Error guessing and fault attacks
- is a technique that should always be used as a complement to other more formal techniques.
- The success of error guessing is very much dependent on the skill of the tester, as good testers know where the defects are most likely to lurk.

Fault attack
- Directed and focused attempt to evaluate the quality, especially reliability, of a test object by attempting to force specific failures to occur.

Exploratory testing
- is a hands-on approach in which testers are involved in minimum planning and maximum test execution.

Structure-based or white-box techniques

Using structure-based techniques to measure coverage and design tests
- Statement Coverage
- Decision Coverage.
- Branch Coverage.

Statement coverage and statement testing
- is a white box test design technique which involves execution of all the executable statements in the source code at least once.
- It is used to calculate and measure the number of statements in the source code which can be executed given the requirements.

Decision coverage and decision testing
- is also known as Branch Coverage.
- Whenever there are two or more possible exits from the statement like an IF statement, a DO-WHILE or a CASE statement it is known as decision because in all these statements there are two outcomes, either TRUE or FALSE.

Other structure-based techniques

Specification-based or black-box technoques

Equivalence partitioning and boundary value analysis Equivalence partitioning
`A black box test design technique in which test cases are designed to xecute representat`
`equivalence partitions. In principle test ases are designed to cover each partition at le`

Equivalence partitioning
- EP is a good all-round specification-based black-box technique.
- Is a black box technique.
- You divide the set of test condition into a partition that can be considered the same

Boundary value analysis
- is the process of testing between extreme ends or boundaries between partitions of the input values.
- Like Start- End, Lower- Upper, Maximum-Minimum, Just Inside-Just Outside values

Decision table testing
`A black box test design technique in which test cases are designed to execute the combinations of inputs and/or stimuli (causes) shown in a decision table.`
- are used to test the interactions between combinations of conditions.
- They provide a clear method to verify testing of all pertinent combinations to ensure that all possible conditions, relationships, and constraints are handled by the software under test.
- If you need to make sure your test cases cover all outcomes in a scenario, read on to learn how to use decision tables.

State transition testing
`A black-box test design technique in which test case are designed to execute valid & invalid state transactions`
- is defined as the software testing technique in which changes in input conditions cause's state changes in the Application under Test.
- is a black box testing technique

Use case testing
- is defined as a software testing technique, that helps identify test cases that cover the entire system, on a transaction by transaction basis from start to the finishing point.

Test management

Test organization

Independent and integrated testing
`A certain degree of Independent often makes the tester more effective at finding defects and failures.`
1. No independent tester, developer will test their own work
2. Independent tester within the development teams.
3. Independent test team or group within the organization.
4. Independent tester from the business organization.
5. Independent test specialist for specific test types such as usability tester, security, performance
6. Independent tester outsourced

Working as a test leader
- Tend to be involved in the planning, monitoring, and control of the testing activities and tasks discussed

expect them to **plan**, **monitor** and **control** the testing work.

Working as a tester
- Involve in **Test planning phase** .

Help to create :
- **Review** and **assesst** requirement
- Help to create **test condition**, **test design**, **test procedure**..

Can **increase** when the phase go on

Defining the skills test staff need
- Application or business domain.
- Technology.
- Testing.

Test planning and estimation

The purpose and substance of test plans
- Writing a test plan guides our thinking.
- The test plan also helps us manage change.
- The test planning process and the plan itself serve as vehicles for communicating with other members of the project team.

test plan is the project plan for the testing work to be done.

It is not a test design specification, a collection of test cases or a set of test procedures;

We find that if we **can explain something in words**, we **understand it**. If not, there's a good chance we don't.

Defining **entry** and **exit** criteria

What to do with your brain while planning tests
- Need to consider the purpose served by the testing work.
- Need to decide how to split the testing work into various levels.
- Need to decide how to best fit your testing work in the level you are responsible for with the testing work done in those other test levels.
- Plan to integrate and coordinate all the testing work to be done with the rest of the project.

Estimating what testing will involve and what it will cost
- break down a testing project into phases.
- Within each phase we identify activities and within each activity we identify tasks and perhaps subtasks.
- To identify we work both forward and backward.
- Backward is consider the risks.
- Estimate the tasks involved as writing test scripts and creating test data..

Estimation techniques

The metrics-based: estimating the test effort based on metrics of former similar projects or based on typical values

The expert-based: estimating the test effort based on the experiece of the owners of the testing tasks or by experts

Factors affecting test effort
- Characteristics of the product
- Characteristics of the development process
- The outcome of testing

Test approaches and strategies
- Deciding on the test approach involves careful consideration of the testing objectives, the project's goals, and overall risk assessment.
  the major types of test strategies
- **Analytical test strategies** have in common the use of some formal or informal analytical technique, usually during the requirements and design stages of the project.
- **Model-based test strategies** : tests are designed based on some model of some required aspect of product such as function, business process, internal structure, non-functional characteristics...
- **Methodical test strategies.**
- **Process- or standard-compliant.**
- **Dynamic strategies**

Exit criteria
- Planed test have been executed
- A defined level of coverage has been achieved
- The number of unresolved defects is within an agreed limit

TEST PROGRESS MONITORING AND CONTROL

Monitoring the progress of test activities
- Give the test team and the test manager feedback on how the testing work is going. Some metrics include: percentage of planned work done
- Provide the project team with visibility about the test results.
- Measure the status of the testing, test coverage and test items against the exit criteria to determine whether the test work is done.
- Gather data for use in estimating future test efforts.

Reporting test status
- Is about effectively communicating our findings to other project stake-holders.
- `provide a summary of the testing performed based on the latest test`

Test control
- Test control is about guiding and corrective actions to try to achieve the best possible outcome for the project. e.g : prioritizing test, change the test schedule, re-evaluating testing work

CONFIGURATION MANAGEMENT

`The purpose of CM is to **establish** and **maintain** the integrity of the component, the testware and their **relationships** to one another through the project and product lifecycle`
- configuration management is in part about determining clearly what the items are that make up the software or system.
- All test items are uniquely identified, version controlled, tracked for changes, related to each other and related to versions of the test items

RISK AND TESTING

Risks and levels of risk
- it's the possibility of a negative or undesirable outcome.
- is determined by **likelihood of the event** and **the impact from that event**

Product risks
- product risk as the possibility that the system or software might fail to satisfy some reasonable customer, user, or stakeholder expectation.

Project risks
- A risk related to management and control of the (test) project, e.g. lack of staffing, strict deadlines, changing requirements, etc.

Tying it all together for risk management
- Deal with test-related risks to the project and product

INCIDENT MANAGEMENT

- Results that vary from expected results.
- An incident report contains a description of the misbehaviour that was observed and classification of that misbehaviour.
- A good incident report is a technical document, clear for its goals and audience, any good report grows out of a careful approach to researching and writing the report.

Tool support for testing

Types of test tool

Understanding the Meaning and Purpose of Tool Support for Testing
- To start with the most visible use, we can use tools directly in testing.
- We can use tools to help us manage the testing process.
- We can use tools as part of what's called reconnaissance, or, to use a simpler term, exploration

Test tool classification
- Test Management Tools
- Requirements Management Tools
- Incident Management Tools (Defect Tracking Tools)
- Configuration Management Tools

Tool support for static testing
- Review Tools
- Static Anaylisis Tools
- Modeling Tools

Tool support for test specification
- Test design tools
- Test data preparation tools

Tool support for test execution and logging
- Test execution tools
- Test harness/unit test framework tools (D)
- Test comparators
- Coverage measurement tools (D)
- Security testing tools

Tool support for performance and monitoring
- Dynamic analysis tools (D)
- Performance-testing, load-testing and stress-testing tools
- Monitoring tools

Tool support for specific application areas
Tool support using other tools
Tool support for specific testing needs

Effective use of tools: potential benefits and risk

Potential benefits of using tools
- reduction of repetitive work;
- greater consistency and repeatability;
- objective assessment;
- ease of access to information about tests or testing.

Risks of using tools
- Unrealistic expectations for the tool
- Underestimating the time, cost and effort for the initial introduction of a tool
- Underestimating the time and effort needed to achieve significant and continuing benefits from the tool;
- Underestimating the effort required to maintain the test assets generated by the tool.
- Over-reliance on the tool, including relying on tools for test design or test execution where manual testing would work better;
- Failing to use proper configuration management and version control facilities for the test assets created for and by the tool;
- Failing to consider and manage issues related to relationships and interoperability between critical tools.
- possibility of the tool vendor going out of business,
- poor or completely non-existent vendor response, whether for support, upgrades, or defect fixes;
- various uncertainties and unforeseen problems, such as the inability to support a new platform.

Special considerations for some types of tools
- Test execution tools
  - Performance testing tools
- Static analysis tools
- Test management tools

Introducing a tool into an organization

Main principles

The organization must be ready to introduce, use and be involved in adapting the tool through its test processes: only an assessment of the organizational maturity could prove its readiness by clearly identifying the strengths, weaknesses and opportunities to do so.

Essentially, the assessment should identify which testing area or processes can be improved by adopting the tool.

Pilot project
The objectives for a pilot project
- to learn more about the tool
- to see how the tool would fit with existing processes or documentation
- I to decide on standard ways of using the tool that will work for all potential users
- to evaluate the pilot project against its objectives

Success factors
Some of the factors contributed to success
- incremental roll-out (after the pilot) to the rest of the organization
- adapting and improving processes, testware and tool artefacts to get the best fit and balance between them and the use of the tool;
- providing adequate support, training, coaching and mentoring of new users;
- defining and communicating guidelines for the use of the tool, based on what was learned in the pilot;
- implementing a continuous improvement mechanism as tool use spreads through more of the organization;
- monitoring the use of the tool and the benefits achieved and adapting the use of the tool to take account of what is learned;
- provide continuing support for anyone using test tools.
- continuous improvement of tool use should be based on information gathered from all teams who are using test tools.