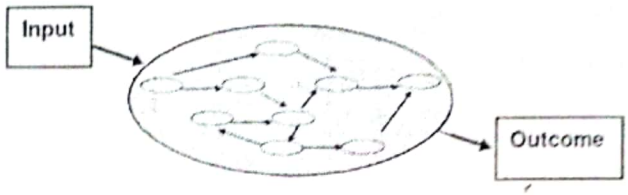
Unit 1

System Testing:

- The goal of system testing is to find defects in features of the system compared to the way it has to be defined in the software system requirements.



- In system testing, both the functional and non functional requirements should be arranged.
- The functional requirements express what the system should be able to do the functionality of the system.
- The non-functional requirements express how the functionality presents itself and behaves.
- Measures of time spent on the testing, on faults found and corrected and on coverage should be collected.
- The system testing must stop when the completion criteria specified in the plan have been met.
- A system test report should be produced when the system testing has been completed.

Acceptance Testing:

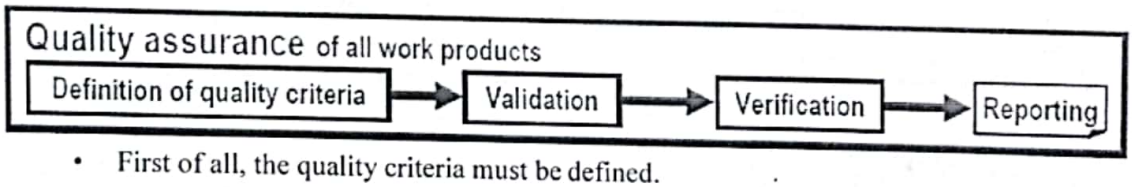
- At the acceptance test level the product is expected to be working and it is presented for acceptance.
- The customer and / or end user must be involved in the acceptance testing.
- There may be a number of acceptance test types, namely
    - Contract acceptance test.
    - Alpha test.
    - Beta test.
- The contract acceptance test also called factory acceptance test. This test must be completed before the product may leave from supplier. The product has to be accepted by the customer.
- An alpha test is usage of the product by representative user at the development size but reflection what the real usage will be like.
- A beta test is usage of the product by selected customers as the customer site.

Supporting Processors:

- Whatever the development model is structured, there will always be a number of supporting activities for the development.
- The primary supporting processes are
  - Quality assurance.
  - Project management.
  - Configuration management.
- The other supporting processes may be
  - Technical writing (production of technical documentation).
  - Technical support (support of environment including tools).

Product Quality Assurance:

- The quality assurance activities must start early and become an integrated part of the entire development project.
- This comprises four activities
  - Definition of quality criteria.
  - Validation.
  - Verification.
  - Quality reporting.



- First of all, the quality criteria must be defined.
- These criteria are the expression of the quality level that must be reached or an expression of "what is sufficiently good".
- There are two quality assurance activities for checking it. The quality criteria have been met by the object under testing namely
  - Validation.
  - Verification.
- Validation is the assessment of the correctness of the product in relation to the user's needs and requirements.
- Validation checks that " Are we building the correct product"
- Verification is the assessment of whether the object fulfils the specified requirements.

- Verification checks that "Are we building the product correctly".
- Quality assurance reports on the findings and results should be produced.

Project Management:

- Project management is the supporting process that takes care of development process, from the first idea to the release.
- The most important activities in project management are;
  - Risk analysis.
  - Estimation.
  - Scheduling.
  - Monitoring and control.
  - Communication.

→ The estimation, risk analysis, and scheduling of the test activities will either have to be done in cooperation with project management or consolidated with the overall project planning.

Configuration Management:

→ The purpose of configuration management is to establish and maintain integrity of work products and product.

→ Configuration management can be defined as

- Unique identification.
- Controlled storage.
- Change management.
- Status reporting.

→ An object under configuration management is called a configuration item.

→ The purpose of identification is to uniquely identify each configuration item and to specify its relatively to the outside world and to other configuration items.

→ The purpose of storage is to ensure that configuration item don't disappear or are damaged.

→ Storage is something physical items that are stored are physically present at a specific place. This place is often called the library or the controlled library.

→ The purpose of change management is to be fully in control of all change requests for a product and of all implemented change.

→ The purpose of status reporting is to make available the information necessary for effective management of the development, testing and maintenance of a product.

Technical Writing:

- ➔ Technical writers are people with special writing skills and education.
- ➔ They assist other staff members which difficult issues need to be made clear to the intended audience in writing.

Product Paradigms:

- ➔ Today we have a number of different product types or product paradigms.
- ➔ A few significant product paradigms are:
    - **System of systems:**
        - A system of systems is the concept integrating existing systems into a single information system with only limited new development.
        - Systems of systems are complicated in nature.
            - From a testing point of view, there are at least three important aspects to take into account when working with systems of systems.
            - System testing of individual systems.
        - Integration testing of systems. Regression testing of system and integration.
    - Examples of systems of systems are:
        - Safety-critical systems.
        - Large mainframe systems.
        - Web-based systems etc.

Metrics and Measurement:

Metric:

A definition of what to measure, including data-type, scale, and unit.

Measuring Method - The description of how we are going to get the data.

Measurements- The actual values collected for the metrics.

We have two types of measurements those are

- ➔ Direct measurements are measurements we get directly from the raw data
- ➔ Indirect measurements are measurements we can calculate from direct measurements.

Test-Related Metrics:

Many measurements can be collected during the performance of the test procedures. A few examples of direct measurements are listed below

Measurements about progress

→ of test planning and monitoring:

a. Tasks commenced

b. Tasks completed

→ of test development

a. number of specified and approved test procedures

b. other tasks completed

→ of test execution and reporting

a. Number of executed test procedures

b. Number of passed test procedures

Analysis And Prsentationof Measurements:

It is never enough to just collect measurements. They must be presented and analyzed to be of real value to us.

Planning Measuring:

When we are goimng to collect data you need to aim for

Agreed metrics: Definitions, scale, and units must be agreed and understood.

Needed measures: What is it you want to know, to monitor and to control.

Repeatable Measurements: Same time of measure and same instrument must give the same measurement.

Creating Confidentiality: Never use measurements to punish or award individuals.

Having a Measurement plan: The plan should outline, what, by whom, when, how, why.

Using the Measurement: Only measure what can be used immediately and give quick and precise feedback.

R. Meghana
III-BCA.

UNIT-II

TESTING PROCESS

Process In General:

A process is a series of activities performed to fulfill a specific purpose. Based on an input to the process and following activities a tangible output is produced.

A process description must always include.

- A definition of the input.
- A list of activities- the procedure.
- A description of output.

For a more comprehensive and more useful process description the following information could also be included

- Entry criteria
- Purpose
- Role
- Methods, Techniques, Tools

Monitoring Process:

It is the responsibility of management in charge of a specific area to know how the pertaining processes are performed. The generic test process consists of five activities. Those are

- Ψ Test Planning
- Ψ Test Analysis and design
- Ψ Test implementation and execution
- Ψ Evaluating exit criteria and reporting
- Ψ Test closure activities

TEST PALANNING AND CONTROL

Input To Test Planning And Control:

- The planning of a test level is based on the relevant test strategy, the project plan for the project, and the master plan.
- The test level plans must be consistent with the master test plan.

Documentation of the Test Planning and Control:

- The documentation of test planning and control controls the following things:

1. Introduction.
2. Test items

3. Features to be tested.
4. Features not to be tested.
5. Approach.
6. Item pass/fail criteria.
7. Test deliverable (work product)
8. Testing tasks.
9. Environmental needs.
10. Responsibilities.
11. Stating and training needs
12. Schedule.
13. Risks.

Activity in Test Planning:

The test planning activity must first of all aim of setting the scene for the testing assignment for the actors in accordance with the frame ware.

The test planning activities are as follows

1. **Defining test object and test basis:**
   a. The object of the testing depends on the test level.
   b. The test planning must identify the test basis and define what it is we are going to test in relation to this. Examples of the most common test basis and corresponding coverage items are

Test Level	Test Basis	Coverage Items
Component Testing	Requirements Detailed Design Code	Statements Decisions Conditions
Component integration Testing	Architectural Design	Internal interfaces Individual parameters Invariants
System Testing	Software requirements specification	Functional and Non-functional requirements
System Integration testing	Product Design	External interfaces Individual parameters Invariants
Acceptance testing	User requirements specification User manual	Use cases Business scenarios
Static Test	Documents the static test is base on	Pages Requirements Test cases

2. **Defining the Approach:**

The test approach must be based on the strategy for the test.

The approach must at least cover the following things.

a. The test methods and test techniques to use

b. The structure of the test specification to be produced and used.

c. The tools to be used.

d. The interface with configuration management.

e. Important constraints, such as availability or "fixed" deadline.

3. Defining the completion criteria:

The completion criteria are what we use to determine if we can stop the testing. The completion criteria are derived from the strategy and should be based on a risk analysis. If the risk is high, then the completion criteria should be strict and if lower the risk then the completion criteria having less demand. The most appropriate completion criteria vary from test level to test level.

Completion criteria for the test may be specified as follows.

a. Specified coverage has been achieved.

b. No known serious faults.

c. The benefits of the system are bigger than known problems.

d. The time has run out.

4. Defining work products and their relationships:

- The no. of deliverables, their characteristics, and estimates of their sizes must be defined, because this is used as input for detailed estimation and scheduling of all the test activities.
- The deliveries or work products from a test level are

    a. Level test plan.

    b. Test Specifications.

    c. Test Environments.

    d. Test logs and journals.

    e. Test reports.

5. Scoping the Test Effort:

- The definition of exhaustive testing is, test case design technique in which for the test case suites comprise all combinations of input values and pre-conditions for component variables.
- We have three mutually dependent parameters that we as testers need to balance.

- The parameters are:

1. Tim: The available calendar time.
2. Resources: The available people and tools.
3. Quality: The quality of the testing.

- These parameters are called quality triangle.
- In a particular project, we need to initially achieve a balance between the time and resources

spenton testing and yore quality of the testing we want.

Work Breakdown Structure:

- When we are going to prepare a test plan, we have to prepare a list of all the tasks to be performed.
- This list should be in the form of a work breakdown structure of the test process.
- A list and a description of every single task must therefore be produced. If a task is not mentioned here it will probably not get done.

Defining Test Roles:

- A project is like a play in which all roles must be filled in order for the play to be performed.
- In my way we have to define roles for each actor in testing process.
- The roles to handle the testing tasks may be
  a. Test leader (manages or responsible)
  b. Test analyst/designer.
  c. Test executer.
  d. Reviewer/inspector.
  e. Domain expert.
  f. Test environment responsible.
  g. (Test) tool responsible.

TEST ANALYSIS AND DESIGN:

- The purpose of the test analysis and design activities is to produce test designs with test conditions and test cases.

1. **Input to Test analysis and Design:**
   a. Test Objectivity.
   b. Scheduling and staffing for the activity.
   c. Definition of test objects.
   d. Approach.
   e. Completion criteria.
   f. Deliverables.

2. **Documentation Of Test Analysis And Design:**

- The results of the test analysis and design should be documented in the test specification. Those are
  a. The test design - also called test groups.
  b. The test cases - many test cases per test design.
  c. Test procedures- often many-to-many relationship with test cases.

Activities in Test Analysis And Design:

The activities in test analysis and design are as follows:

Test Analysis and Design:

- The purpose of the test analysis and design activities is to produce test design with test conditions.

Activities in test analysis and design:

1. **Defining Test Designs**:

- In test design, the testing task is broken into a number of test design or test group.
- A test design or test group specifications should have the following contents.

a. Features to be tested.

b. approach retirement

c. list of high-level test cases

d. list of expected test procedures

e. feature pass/fail criteria

2. **Identification of Test Conditions**:

- The features to be tested mentioned in the test design can be expressed as test conditions or test requirements.
- A test condition is a verifiable item or element.
- The test conditions are based on or identical to our coverage items.
- They are the items we are covering when we test the test objects.

Creation of Test Cases:

- Based on the test conditions, we can now produce our first high-level test cases and subsequently low-level test cases.
- A high level test case is a test case without specific values for input data and expected results, but with logical operations or other means of defining what to test in general terms.
- The test cases we design should follow these things more are

a. Effective: have a reasonable probability of detecting errors.

b. Exemplary: be practical and have a low redundancy.

c. Economic: have a reasonable development cost and return on investment.

d. Evolvable: be flexible, structured, and maintainable.

From the high-level test cases we go on to define the low-level test cases.

A low-level test case is a test case with specific values defined for both input and expected result.

Requirements:

- The purpose of this is to make testers understand requirements better, and equip them to take part in the worth with the requirements and to express test-related requirements for the requirements produced for a product.

Requirement Levels:

- We have three levels in requirements.

  1. Business requirements.
  2. User requirements.
  3. System requirements.

Requirement Types:

- we have two types of requirements those are:

  1. Function requirements.
  2. Non-functional requirements.

- The functional and non-functional requirements together from the product quality requirements.

Requirement Styles:

- requirements can be expressed in many ways those are

  1. Statements.
  2. Tasks
  3. Models.
  4. Tables.

- The most common style is the statement styles. Here each requirement is expressed as a single sentence in natural language.

Traceability:

- Traces should be two-way. Those are 'backward' trace and 'forward' trace.
- The 'backward' trace is also very helpful if you need to identify which coverage items a test case is covering.
- The 'forward' trace will also make it possible to quickly identify the test case.

Metrics For Analysis And Design:

- Number of specified test conditions and high-level requirements over time.

- Number of defects found during analysis and design.

TEST IMPLEMENTATION AND EXECUTION:

The purpose of the test implementation is to organize the test cases in procedures and to perform the physical test in the correct environment.

Activities in Test Implementation And Execution:

1. **Organizing Test Procedures**:

   - The low-level test cases should how be organized and assembled in test procedures and/or test scripts.
   - The term "procedure" is mostly used when they are prepared for manual test execution while the term "script" is mostly used for automatically executable procedures.
   - the documentation of a test procedure must at least include

     1. Unique identification.
     2. Description.
     3. References to high-level test cases.

2. **Test Environment Specification and Testing**:

   - The test environment is a necessary prerequisite for the test execution without an proper environment the test is either not executable at all or the results will be open to doubt.
   - The description of the test environment must be a specific as possible in order to get the right test environment - established at the right time.
   - the descriptions of the test environment must cover
     1. Hardware
     2. Software
     3. Peripherals (printers.fax, CD reader, etc...)
     4. Network
     5. Tools and utilities
     6. Data
     7. Physical environment (room, furniture etc...)
     8. Communication (phones, internet etc...)

3. **Checking Execution Entry Criteria**:

   - If the test object has not passed the entry criteria defined for it, do not start the test execution.
   - Efficient and timely execution of the tests is dependent on the support process being in place.
   - It is important that the configuration management is working well.

4. **Test Execution**:

- In structured testing, in principle all the testers have to do during test execution is to follow the test specification and register all incidents on the way.
- We have taken great care in writing the test procedures; there are several reasons for this. those are

1. We need to be able to collect actual time spent and compare it with the estimates to improve our estimation techniques.

2. We need to be able to compare the progress with the plan.

3. It should be possible to make a complete audit of the test.

5. **Identifying Failures**:

- For each test case we execute the actual results should be logged and compared to the expected result, defined as part of the test case.
- We need to be very careful when we compare the expected result with the actual result, in order not to mix failures or report correct behavior as failures.
- Any failure must be reported in the incident management system.

6. **Test Execution Logging**:

- As we execute, manually or by the use of tools, we must log what is going on.
- We must record the precise identification of what we are testing and the test environment and test procedures we use.
- We must also log the result of the checking.
- The IEEE829 standard suggests the following contents of a test log
  1. Description of the test
  2. Activity and event entries.

7. **Configuration Testing and Regression Testing:**

- **Configuration Testing**:

1. Configuration testing is the first to be performed after detect correction.
2. It is done to ensure that the defect has indeed been successfully removed.
3. The test that originally unrevealed the defect by causing a failure is executed again and this time it should pass without problems.
4. This is illustrated by the dark rectangle in the place where the defect was.

- **Regression Testing**:

a. Regression testing is repetition of tests that have already been performed without problems to ensure that defects have not been introduced or uncovered as a result of the change.

b. The amount of regression testing depends on issues such as

1. The risk involved.
2. The architecture of the system or product.
3. The nature of the defect that has been corrected.

- **METRICS For Implementation And Execution:**

1. No. of created test environments overtime.
2. No. of created test procedures overtime.
3. No. of passed test procedures overtime.
4. No. of test procedures run for regression testing overtime.

EVALUATING EXIT CRITERIA AND REPORTING:

a. Test execution, recording, control, testing and regression test must be continued until we believe that the exit criteria have been achieved.

b. Activities in test progress and completion reporting:

Checking For Completion:

A check against the test exit criteria is mandatory before we can say that the testing is completed at any level. Examples of exit criteria are

a. Specified coverage has been achieved.
b. Specified no. of failures found per test effort has been achieved.
c. No known serious faults.

Metrics for Progress and Completion Reporting:

1. No. of tasks commenced overtime.
2. Task completion percentage overtime.
3. No. of task completed overtime.
4. Time spent in each task overtime.

TEST CLOSURE:

Activities in Test Closure:

1. Check Completion Again

a. Before we definitively close the door to the testing assignment. We need to make extra sure that we have met the part of the exit criteria.

2. Delivering and Archiving Test Ware

A. The test ware we have produced are valuable aspects for the organization and should be handled carefully.

B. If the organization has a well-working configuration management system. This is what we must use to safeguard the test ware.

C. If such system does not exist, we must arrange with those who are taking over

responsibility for the product how the test ware must be secured.

3. Retrospective Meeting

a. The last thing we have to do is to report the experiences we have gained during our testing.
b. The measurements we have collected should be analyzed and any other experiences collected and synthesized as well.
c. It is important that we as testers finish our testing assignment properly by producing an experience report.

- **Metrics For Test Closure Activities**:
  Metrics to be defined for these activities may include number of tasks commenced overtime, task completion percentage overtime etc..

Test Management

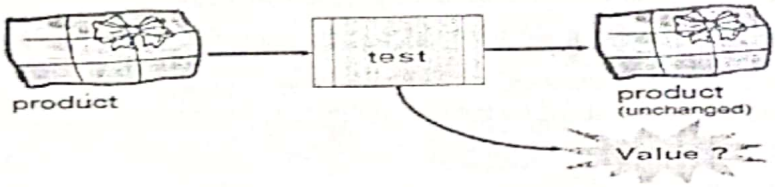
UNIT-III

INTRODUCTION:

- Test management is the art of planning and directing a test assignment to success.
- In many ways like project management, and yet not quite the same.
- Test management must be done in close cooperation with project management,
- sometimes by the same person,
- Sometimes by different people.
- The test manager is the link between the test team and the development team

Business Value of Testing :

1. Purpose of Testing
2. Testing business case
    - The Value of Product Improvement
    - The Value of Decision Improvement
    - The Value of Process Improvement
3. On the face of it, testing adds no value.
4. The product under testing is—in principle—not changed after the test has been executed.



- The business value of testing lies in the savings that the organization can achieve from improvements based on the information the testing provides.
- Improvements can be obtained in three places:
    - The product under development
    - The decisions to be made about the product
    - The processes used in both testing and development
- Sometimes be difficult to understand and express what the value is
- Test managers know and understand the value of testing
- Test managers communicate to other project participants, and to higher management.

Purpose of Testing:

- Purpose of testing is getting information about the product under testing.
- Testing is the intelligence office of the company
- The places we gather our raw data from are the test logs and the incident reports
- A few examples of such information are:
    - Number of passed test cases
    - Coverage of the performed test
    - Number and types of failures

- Defects corrected over time
- Root causes of the failures

- Most of this information is "invisible" or indigestible unless we testers make it available in appropriate formats

The Testing Business Case:

- A well-established way to express the value of testing for the product is based on the cost of quality.
- This can be expressed as value of product improvement:

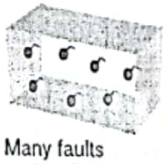
Value of product improvement =
(cost of failure not found – cost failure found) – cost of detection
To this we can add
Value of decision improvement =
(cost of wrong decision – cost of right decision) – cost of getting decision basis
Value of process improvement =
(cost using old process – cost using better process) – cost of process improvement

- A value may be expressed either quantitatively or qualitatively.
- Quantitative values can be expressed in actual numbers— euros, pounds, or dollars or numbers of something,
- Qualitative values cannot be calculated like that, but may be expressed in other terms or "felt."

The Value of Product Improvement:

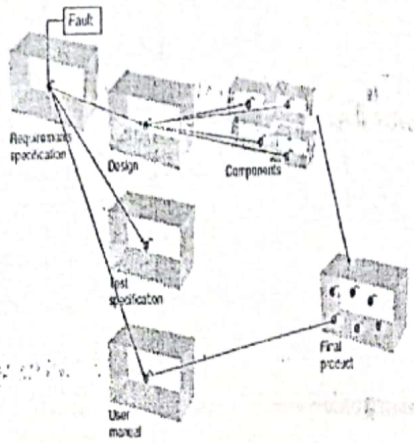
- The value of product improvement is the easiest to assess.
- One goal of all development is reliability in the products we deliver to the customers.
- Reliability is the probability that software will not cause the failure of a system for a specified time under specified conditions.
- A product's reliability is measured by the probability that faults materialize in the product when it is in use.

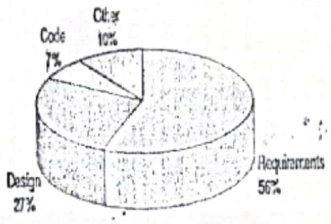
No faults
= 100% reliability



Many faults
= x% reliability

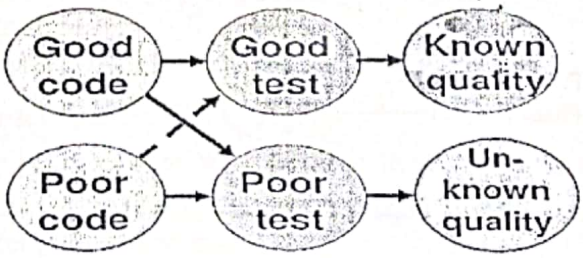
- The less failures that remain in the product we release, the higher is the reliability of the product and the lower the risk of the product
- The earlier we get a defect removed the cheaper it is.
- Reviews find defects and dynamic testing finds failures, and this enables the correction of the underlying defects.
- The cost of the defect correction depends on when the defect is found.
- If the defect remains in the product and is not found until encountered as a failure in dynamic test, it costs 100 units to correct it.
- The failures found during development and testing are called internal failures, and they are relatively cheap.
- If the customer gets a failure in production—an external failure
- The basic reason for this raise in cost is that defects in software do not go away if left unattended; they multiply
- To get the full value of the test, it should start as early as possible in the course of a development project, preferably on day 1





The Value of Decision Improvement:

- The point of view of decision making such as decisions concerning release (or not) of a product the confidence in the product and quality of the decisions are proportional to the quality and the amount of the information provided by testing
- As testing progresses, more and more information is gathered and this enhances the basis for the decisions.
- The more knowledge the decision makers have about
  - Parts of the product have been tested
  - Depth coverage of test
  - Detected defects have been removed
  - Still remaining
- The quality of the testing reflects in the quality of the information it provides.
- Good testing provides trustworthy information and poor testing leave us in ignorance
- If the starting point is a good product, a good test will provide information to give us confidence that the quality is good.
- If the starting point is a poor product, a good test will reveal that the quality is low.
- A test report with documentation of the test and the test results can be used to prove that we fulfilled contractual obligations



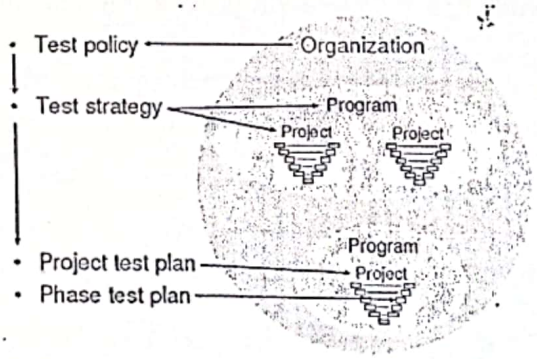
The Value of Process Improvement:

- The process improvement point of view the information gained from testing is invaluable in the analysis of how well processes fit and serve the organization.
- The results of such analysis can be used to identify the process that could be the subject for process improvement.
- The process to improve may be both the testing process and other processes.
- When the testing process improves, the number of failures sent out to the customer falls.
- The organization's reputation for delivering quality products will rise.

Test Management Documentation:

- **Overview**

- **Higher Management Documentation**
  - Test Policy
  - Test Strategy
- **Project Level Test Management Documentation**
  - Master Test Plan
  - Level Test Plan
  - Test Plan Template
  - Scheduling Test Planning
  - Test Report
- Proper test management requires that information about the decisions that test management makes is available and comprehensive to all stakeholders.
- Decisions are normally captured in a number of documents.
- The test management documentation comprises:
  - Test policy
  - Test strategy
  - Project test plan
  - Level test plan
- The **test policy** holds the organization's philosophy toward software testing.
- The **test strategy** is based on the policy. It can have the scope of an organizational unit or a program It contains the generic requirements for the test for the defined scope.
- A **master test plan** is for a particular project. It makes the strategy operational and defines the test levels to be performed and the testing within those levels.
- A **level test plan** is for a particular test level in a particular project. It provides the details for performing testing within a level.



- The presentation of this documentation depends on the organization's needs, general standards, size, and maturity.
- It can also vary from all the information being presented together
  - In one document, or even as part of a bigger document, to it being split into a number of individual documents

Higher Management Documentation:

- Higher management, that is management above project managers and test managers, is responsible for the two types of test management documentation.
- The documentation is used by everybody in the organization involved in testing.

Test Policy:

- The test policy defines the organization's philosophy toward software testing.
- It is the basis for all the test work in the organization.

- A policy must be behaviour-regulating in the good way—it is like a lighthouse for all the testing activities
- The test policy defines the organization's philosophy toward software testing.
- It is the basis for all the test work in the organization
- The test policy must be short and to the point.
- It is the responsibility of the top management to formulate the policy
- The test policy must include:
  - 1. Definition of testing
  - 2. The testing process to use
  - 3. Evaluation of testing
  - 4. Quality targets
  - 5. Approach to test process improvement
- The test policy applies to all testing. The policy must cover all test targets.
- This means that there must be a policy for:
  - » Testing new products
  - » Change-related testing
  - » Maintenance testing
- Test Policy; Definition of Testing
- The definition of testing is a brief statement formulating the overall purpose of the test in the organization.
- Test Policy; The Testing Process
- The testing process is an overview of the activities to be performed or a reference to a full description of the testing process.
- Test Policy; Evaluation of Testing
- The evaluation of testing is the measurement to be made in order for the quality of the testing to be determined.
- Test Policy; Quality Targets
- The quality targets to be achieved should be expressed so that the measurements can be used to see if we reach the goals.
- Test Policy; Approach to Test Process Improvement

Test Strategy:

- The test strategy is high-level, and it should be short.
- It should also be readily available to all with a stake in the testing within the scope of the strategy.
- The strategy could be issued in a document, but it would be a good idea to present it in table form on a poster or on the intranet in the organization.
- The overall test strategy may be chosen among the following possible approaches to the testing:
  - Analytical—Using for example a risk analysis as the basis
  - Model-based—Using for example statistical models for usage
  - Consultative—Using technology guidance or domain experts
  - Methodical—Using for example checklists or experience
  - Heuristic—Using exploratory techniques
  - Standard-compliant—Using given standards or processes
  - Regression-averse—Using automation and reuse of scripts
- A test strategy for a defined scope could contain the following information:
- Test strategy identifier

- 1. Introduction
- 2. Standards to use
- 3. Risks to be addressed
- 4. Levels of testing and their relationships
- For each level, as appropriate
- 4.1 Entry criteria
- 4.2 Exit criteria
- 4.3 Degree of independence
- 4.4 Techniques to use
- 4.5 Extent of reuse
- 4.6 Environments
- 4.7 Automation
- 4.8 Measurements
- 4.9 Confirmation and regression testing
- 5. Incident management
- 6. Configuration management of testware
- 7. Test process improvement activities

- The strategy identifier is the configuration management identification information
- for the strategy itself. It could be formed by the:
- » Name of the strategy
- » Organizational affiliation
- » Version
- » Status

1. Strategy-Introduction:

- Purpose of the document
- Scope of the strategy
- References to other plans, standards, contracts, and so forth
- Readers' guide

2. Strategy-Standards to Be Complied With:

- Standards may be both external to the organization and proprietary standards.
- Standards are very useful. Many people with a lot of experience have contributed to standards.

3. Strategy- Risks:

- The basis for the strategy can be the product risks to mitigate by the testing. Appropriate project risks may also be taken into consideration.

4. Strategy-Test Levels and Their Relationships

Description of the test levels:

The levels can for example be:

1. Component testing
2. Component integration testing
3. System testing

System integration testing

Acceptance testing

Strategy- Level Entry Criteria:

- This is a description of what needs to be in place before the work in the test level can start.
- The strictness of the entry criteria depends on the risk: The higher the risk the stricter the criteria.

6. Strategy- Level Exit Criteria:

- The testing exit or completion criteria are a specification of what needs to be achieved by the test. It is a guideline for when to stop the testing—for when it is "good enough."
- Testing completion criteria represent one of the most important items in a comprehensive test strategy, since they have a great influence on the subsequent testing and the quality of a whole system.
- Testing completion criteria represent one of the most important items in a comprehensive test strategy, since they have a great influence on the subsequent testing and the quality of a whole system.
- The test report has been approved by the project manager.

7. Strategy-Degree of Independence:

- The degree of independence increases with the "distance" between the producer and the tester. These degrees of independence in testing have been defined:
  - 1. The producer tests his or her own product
  - 2. Tests are designed by another non tester team member
  - 3. Tests are designed by a tester who is a member of the development team
  - 4. Tests are designed by independent testers in the same organization
  - 5. Tests are designed by organizationally independent testers (consultants)
  - 6. Tests are design by external testers (third-party testing

8. Strategy-Test Case Design Techniques to Be Used:

- The choice of test case design techniques is very much dependent on the risk—high risk: few, comprehensive techniques to choose from; low risk: looser selection criteria

9. Strategy- Extent of Reuse:

- Reuse can be a big money and time saver in an organization.
- Effective reuse requires a certain degree of maturity in the organization.
- Configuration management needs to be working well in order to keep track of items that can be reused.

10. Strategy- Environment in Which the Test Will Be Executed:

- Generic requirements for the test environment must be given here.
- The specific environment must be described in the test plan, based on what the strategy states

11. Strategy-Approach to Test Automation:

- This is an area where the strategy needs to be rather precise in order for tool investments not to get out of hand.
- Technical people, testers—love tools. Tools are very useful and can ease a lot of tedious work.

12. Strategy-Measures to Be Captured:

- Measures are also necessary to be able to monitor and control the progress of the testing.
- We need to know how the correspondence is between the reality and the plan.
- We also need to know if and when our completion criteria
- have been met.

13. Strategy- Approach to Confirmation Testing and Regression Testing:

- Confirmation testing is done after fault correction to confirm that the fault has indeed been removed.
- Regression testing should be done whenever something has changed in the product

14. Strategy-Approach to Incident Management:

- The configurations management system is sufficient here
- It must be described how incidents are to be reported and who the incident reports should be sent to for further handling.

15. Strategy-Approach to Configuration Management of Testware:

- Configuration management of testware is important for the reliability of the test results
- A good configuration management system will also help prevent extra work in finding or possibly remaking testware that has gone missing— something that happens all too often in testing

16. Process Strategy- Approach to Test Improvement:

- This could be a refinement of the approach described in the policy

Project Level Test Management Documentation:

The two types of test management documentation discussed in this section belong to a particular project.

- The master test plan should be produced by the person responsible for testing on the project, ideally a test manager.
- The level test plans should be produced by the stakeholder(s) carrying the appropriate responsibility.

Master Test Plan:

- The master test plan documents the implementation of the overall test strategy for a particular project.
- The master test plan must be closely connected to the overall project plan, especially concerning the schedule and the budget.
- The master test plan has many stakeholders and missions, and it must at least provide the information indicated in the following list to the main stakeholders

Stakeholder	Information
All	Test object = scope of the test for each level
	Involvement in the testing activities
	Contribution to the testing activities
	Relevant testing deliverables (get/produce)
Management	Business justification and value of testing
	Budget and schedule
	Test quantity and quality
Development	Expectation concerning delivery times
	Entry criteria for deliverables
Test team	Test levels
	Schedule
	Test execution cycles
	Suspension criteria and exit criteria
Customer	Test quantity and quality

Level Test Plan:

A level test plan documents a detailed approach to a specific test level, for example a component test or acceptance test.

- The level test plan describes the implementation of the master test plan for the specific level in even more precise detail.
- The size of a level test plan depends on the level it covers—a component test plan for single components may be just 5–10 lines; system test plans may be several pages.

Test Plan Template:

- The structure of the test plans, both the master test plan and any level test plans, should be tailored to the organization's needs.
- A template could be based on the IEEE 829 standard. This standard suggests the following contents of a test plan:
    1. Introduction (Scope, risks, and objectives)
    2. Test items (or) Test objects
    3. Features (quality attributes) to be tested
    4. Features (quality attributes) not to be tested
    5. Approach (targets, techniques, templates)
    6. Item pass/fail criteria (exit criteria including coverage criteria)
    7. Suspension criteria and resumption requirements
    8. Test deliverables (work products)
    9. Testing tasks
    10. Environmental needs
    11. Responsibilities
    12. Staffing and training needs
    13. Schedule
    14. Risks and contingencies
- The test plan identifier is the configuration management identification information for the test plan itself. It could be formed by the
- » Name of the plan
- » Organizational affiliation
- » Version
- » Status

Scheduling Test Planning:

- Planning is important and planning takes time
- It is important to plan activities rather than just jump headfirst into action.
- The work on the planning provides a deeper understanding of the task at hand you have written down or sketched out on a piece of paper.
- Because planning takes time and because it is important, it should be planned so that it can start as early as possible
- The benefits of starting test planning early are many:
    - » There is time to do a proper job of planning.
    - » There is more time to talk and/or negotiate with stakeholders.
    - » Potential problems might be spotted in time to warn all the relevant stakeholders.
    - » It is possible to influence the overall project plan.
- When you plan you have to keep in mind that a plan needs to be SMART:
    - Specific—Make it clear what the scope is
    - Measurable—Make it possible to determine if the plan still holds at any time

- Accepted—Make every stakeholder agree to his or her involvement
- Relevant—Make references to additional information; don't copy it
- Time-specific—Provide dates

- The test plan should be reviewed and approved by all stakeholders to ensure their commitment

Test Report:

- The purpose of test reporting is to summarize the results and provide evaluations based on these results.
- A test report should be issued at the completion of each test level and the end of the entire testing assignment task
- According to IEEE 829 the test report should contain:
- Test report identifier

1. Summary
2. Variances
3. Comprehensiveness assessment
4. Summary of results
5. Evaluation
6. Summary of activities

Test Estimation:

- General Estimation Principles
- Test Estimation Principles
- The Estimation Process
- Estimation Techniques
  - Estimation; Best Guess (FIA)
  - Estimation; Analogies and Experts
  - Estimation; Delphi Technique
  - Estimation; Three-Point Estimation
  - Estimation; Function Points
  - Estimation; Test Points
  - Estimation; Percentage Distribution
- From Estimations to Plan and Back Again
- Get Your Own Measurements

General Estimation Principles:

- Estimation is a prediction of how much time it takes to perform an activity.
- It is an approximate calculation or judgment.
- An estimate is typically based on the professional understanding of experienced practitioners.
- Estimation is input to the scheduling. Only in that activity will we transform the estimated hours into dates.
- Estimates are predictions about the future and predictions are by definition uncertain.
- You should always calculate with an uncertainty in every estimate and document this uncertainty with the estimate

Test Estimation Principles:

- Estimating test activities is in many ways like all other estimation in a project.
- We need to take all tasks, even the smallest and seemingly insignificant, into account.

The time to complete must be estimated for each task defined in the task section, including all the test process activities from test planning to checking for completion.

The test estimation is different from other project estimations, because the number of failures is not known in advance

- The estimation must include:
  - Time to produce incident registrations
  - Possible time to wait for fault analysis
  - Possible time to wait for fault correction
  - Time for retest and regression test (minimum three iterations!)

The Estimation Process:

- You should of course use your organization's standard process for estimation, if there is one.
- Estimation procedure like the generic one described here.
- 1. Define the purpose of the estimation—Is this estimation the first approach, for a proposal, or for detailed planning?
- 2. Plan the estimating task—Estimation is not a two-minute task; set Sufficient time aside for it.
- 3. Write down the basis for the estimation—Here the scope and the size of the work are determined, and all factors that may influence the estimates are registered. This includes factors related to the nature of the processes we are working by, the nature of the project we are working in, the people we are working with, and any risks we are facing.
- 4. Break down the work—This is the work breakdown (i.e., the listing of all the tasks to estimate). Do this as well as possible in relation to the purpose.
- 5. Estimate—Use more than one technique as appropriate.
- 6. Compare with reality and re-estimate—This is the ongoing monitoring and control of how the work that we have estimated is actually going.

Estimation Techniques:

- The following estimation techniques are the most used and an expression of the best practice within estimation.
  - » FIA (finger in the air) or best guess
  - » Experience-based estimation
  - » Analogies and experts
  - » Delphi technique
  - » Three-point estimation (successive calculation)
  - » Model-based estimation
  - » Function points
  - » Test points
  - » Percentage distribution

Estimation-Best Guess (FIA):

- This technique is more or less pure guesswork, but it will always be based on some sort of experience and a number of (unconscious) assumptions.
- The uncertainty contingency is probably around 200%–400% for estimates based on best guess.

Estimation; Analogies and Experts:

- a testing project that is comparable to the one we are estimating, we might use that as a baseline to do our estimation.
- Analogies may also be based on metrics collected from previous tests.
- We may estimate the number of iterations of the test based on recent records of comparable test efforts.

- We can calculate the average effort required per test on a previous test effort and multiply.
- Experts, in the estimation context, know what they are talking about and have relevant knowledge

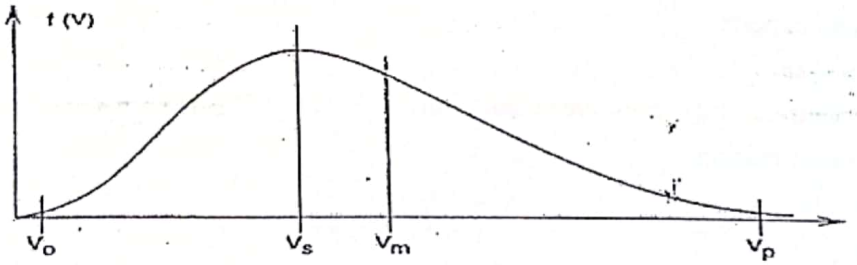
Estimation; Delphi Technique:

- This is a simple technique even in complex situations.
- You must appoint an estimation group as appropriate.
- This can be stakeholders and/or experts in the tasks to estimate.
- The steps in this estimation process are:
  - Each member of the group gives an estimate.
  - The group is informed about the average and distribution of the estimates.
  - Those giving estimates in the lower quartile and in the upper
  - quartile are asked to tell the rest of the group why their estimates
  - were as they were.
  - » The group estimates again—this time taking the previous result and the provided arguments for the "extreme" estimates into account.
  - » This may continue two, three, four, or more times until the variation in the estimates is sufficiently small.

Estimation; Three-Point Estimation:

- Three-point estimation is a statistical calculation of the probability of finishing within a given time.
- The technique is useful for quantifying uncertainty to the estimate.
- The technique is also called successive calculation because tasks are broken down and the estimates successively calculated until the variance is within acceptable limits.
- Three point estimation is based on three estimates:
  - The most optimistic time (ideal conditions)
  - The most likely time (if we do business as usual)
  - Thea most pessimistic time (Murphy is with us all the way)

From these three estimates it is possible to define the distribution function for the time to finish. It could look like the figure shown where V_o = most optimistic; V_s = most likely; V_p = most pessimistic; and V_m = mean.



- We can use the approximated formula to derive:
- $Vm = (Vo + 3*Vs + Vp) / 5$
- $S = (Vp - Vo) / 5$ (the standard deviation)
- we can calculate the time needed for any probability
- of finishing the task

Estimation; Function Points:

- This technique is a factor estimation technique initially published by Albrecht in 1979.
- it now maintained by IFPUG —International Function Points User Group.
- The estimation is based on a model of the product.
- Five aspects of the product are counted from the model:

- External inputs
- External outputs
- External enquiries
- Internal logical files
- External interface files

- The counts are then multiplied with a weight and the total of the weighted counts is the unadjusted sum.
- It requires some training to be able to count function points correctly.
- The disadvantage of using function points is that they require detailed requirements in advance

Estimation; Test Points:

- In 1999 Martin Pol et al. published a dedicated test estimation technique called test points as part of the TMAP method.
- The technique is based on the function point technique, it provides a unit of measurement for the size of the high-level test
- The technique converts function points into test points based on the impact of specific factors that affect tests, such as:
  - » Quality requirements
  - » The system's size and complexity
  - » The quality of the test basis (the document(s) the test is specified toward)
  - » The extent to which test tools are used

Estimation; Percentage Distribution:

- This technique is a so-called top-down estimation technique.
- Test efforts can be derived from the development effort.
- The estimation using this technique starts from an estimate of the total effort for a project.
- This estimate may be the result of the usage of appropriate
- Estimation techniques at the project management level.
- The next step is to use formulas (usually just percentages) to distribute
- This total effort over defined tasks, including the testing tasks
- The formulas are based on empirical data, and they vary widely from organization to organization.

Estimation; Percentage Distribution:

Activity	%
Requirements	9.5
Design	15.5
Coding	20
Test (all test phases)	27
Project management	13
Quality assurance	0
Configuration management	3
Documentation	9
Installation and training	3

All phases	%
Component testing	16
Independent testing	84
	100
Independent testing	%
Integration testing	24
System testing	52
Acceptance testing	24
	100
System testing	%
Functional system testing	65
Nonfunctional system testing	35
	100

From Estimations to Plan and Back Again:

- The estimation is done to provide input to the scheduling activity in the project planning.
- In the scheduling we bring the estimates for the defined testing tasks together with the people, who are going to be performing the tasks.
- Estimations should be in hours.
- The scheduling provides the dates: dates for when the performance of each of the tasks should begin, and dates for when they are expected to be finished.
- When defining the expected finish date for a task we need to take several aspects into account:
    - The start and/or finish dates of others tasks that this task depends
    - on to start, if any
    - The earliest possible start date for the task
    - The general calendar regarding public holidays
    - The pure estimate for the time to finish the task
    - The efficiency of the employee(s) to perform the task—typically 70–
    - 80% for a full time assignment
    - The employee(s)'s availability—this should NOT be less than 25%

Get Your Own Measurements:

- All estimates are based on experience—maybe very informally (FIA), maybe very formally (like function points).
- Better estimation means more reliable estimations
- we both, management and customers, want.
- In order to get better estimates we need to collect actual data.
- Empirical data for estimation is part of the measurements we are collecting.
- So we need to chip in to establish a set of simple measurements of
    - time,
    - costs, and
    - size in all project

Test Progress Monitoring and Control:

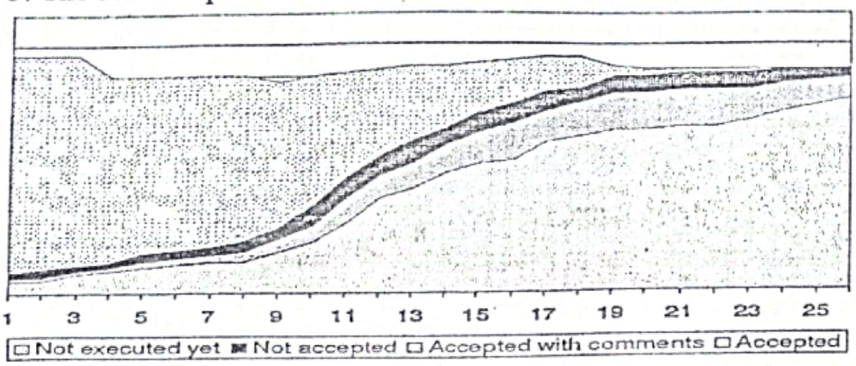
- **Collecting Data**
- **Presenting the Measurements**
    - S-Curves
    - Pie Chart
    - Check Sheets
    - Risk-Based Reporting
    - Statistical Reporting
- **Stay in Control**

S-Curves:

- The most used, most loved, and most useful way of presenting progress information and controlling what's happening is S-curves.
- S-curves can be used for many different metrics
- for example, be:
    - Test cases (run, attempted, passed, completed)
    - Incidents (encountered, fixed, retested)
- S-curves can give us early warnings of something being wrong

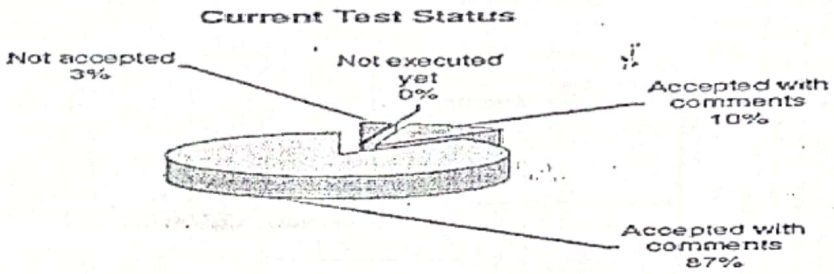
The principle in S-curves is that our work falls in three phases:

- Phase 1: Slow start—not more than 15–25%
- Phase 2: Productive phase—55–65%
- Phase 3: The difficult part—10–25%



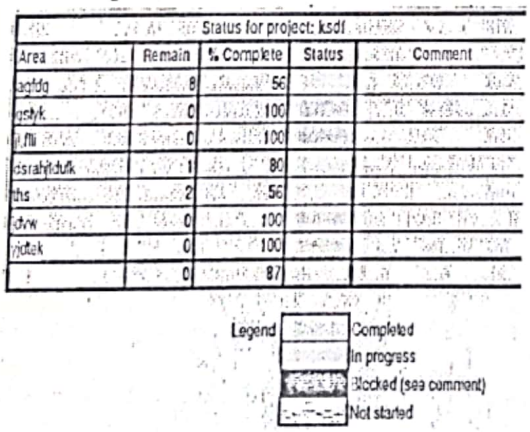
Pie Chart :

- Pie charts are used to give an overview of the proportions of different aspects relative to each other.



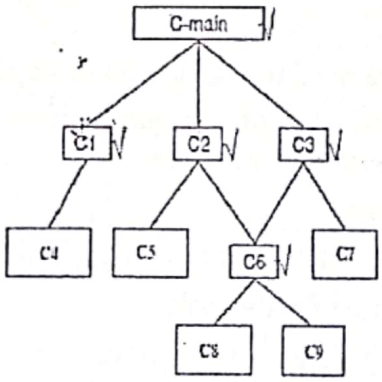
Check Sheets:

- Check sheets are a good way to give a quick overview of status. They can be used to show progress compared to plan.
- Check sheets can be presented as colorful graphics or expressed as lists or hierarchies.



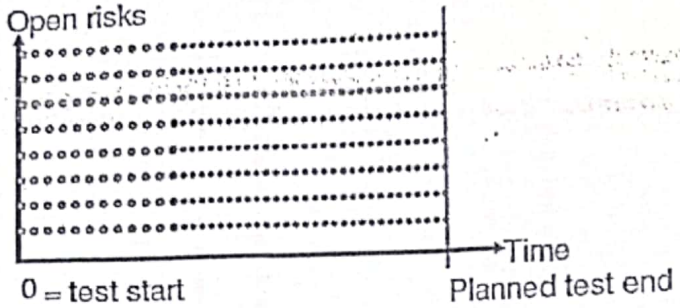
Status for project: ksdf				
Area	Remain	% Complete	Status	Comment
agfdg	8	56		
gstyk	0	100		
ufffi	0	100		
risrahjtdufk	1	80		
ths	2	56		
dvw	0	100		
yjdtek	0	100		
	0	87		

Legend: Completed; In progress; Blocked (see comment); Not started



Risk-Based Reporting:

- If our test approach is based on identified and analyzed risks it is appropriate to report on the test progress in terms of these risks.
- The purpose of this test is to eliminate the risk,



Statistical Reporting:

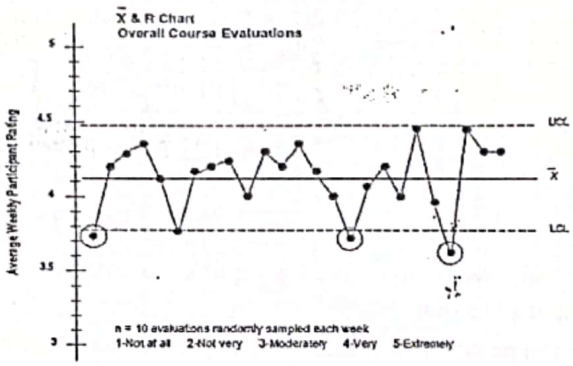
Statistics is the science of patterns in a variable world. We can say that statistics make the invisible visible.

This means that statistical methods can be used to help us:

» Understand the past

» Control the present

» Predict the future



Stay in Control:

- Sometimes we need to take action to stay in control.
- Keeping the triangle of test quality in mind, you have three aspects we can change—and you must change at least two at the time.
- The aspects are:
  - The resources for the task
  - The time for the task
  - The quality of the work to be performed
- Usually when things are getting out of control it is because we are behind schedule or because our time frame has been squeezed.
- The important message to the test manager is
- If you do not control the test, it will control you!

Testing and Risk:

- **Introduction to Risk-Based Testing**
  - Risk Definition
  - Risk Types
    - Project Risks
    - Product Risks

- Testing and Risk Management

Risk Management

- Risk Identification
- Lessons Learned and Checklists
- Risk Workshops
- Brainstorming
- Expert Interviews
- Independent Risk Assessments

- **Risk Analysis**
  - Risk Template
  - Perception of Risks
  - Scales for Risk Analysis
  - Effect
  - Probability
  - Risk Level
- **Risk Mitigation**
  - What to Do to Mitigate Risks
  - How to Mitigate Risks by Testing
  - When to Mitigate Risks by Testing
- The golden rule of testing is:
  - **Always test so that whenever you have to stop you have done the best possible test.**
- The best possible test depends on the risk associated with having defects left in the product when it is released to the customer
- It is impossible to test everything. There is a risk involved in all sample control: the risk of overlooking defects in the areas we are not testing.
- **Risk Definition:** The possibility of realizing an unwanted negative consequence of an incident.
- A risk therefore has two aspects:
  - » Effect (impact—consequence)
  - » Probability (likelihood—frequency)
- The two aspects of risk can be combined in
  - Risk level = probability x effect

Risk Types:

- Risks are divide them into classes, corresponding to where they may hit, or what they are threatening.
- Risks hit in different places, namely;
- **The business**: The business risks are things threatening the entire company or organization from a "staying-in-business" point of view.
- **The processes:** Process risk threatens the effectiveness and efficiency with which we work on an assignment. Like
  - Missing processes
  - The organization's lack of knowledge about the processes
  - Inadequate processes
  - Inconsistencies between processes
  - Unsuitable processes

- Lack of support in the form of templates and techniques

- **The project:** Project risks are related to the project and the successful completion of the project. Risks concerning the project may be originated in:
  - People assigned to the project (e.g., their availability, adequate skills and knowledge, and personalities)
  - Time
  - Money
  - Development and test environment, including tools
  - External interfaces
  - Customer/supplier relationships
- The product: Product risks are related to the final product. Product risks may be originated in:
  - Functional and nonfunctional requirements
  - Missing requirements
  - Ambiguous requirements
  - Misunderstood requirements
  - Requirements on which stakeholders do not agree

Testing and Risk Management:

- Testing and management of risks should be tightly interwoven as they support each other.
- Testing can be based on the results of risk analysis, and test results can give valuable feedback to support continuous risk analysis.
- The risk analysis results can also be used to prioritize and distribute the test effort.
- Testing can also mitigate project risks

Risk Management:

- Risk management consists of the following activities:
- » **Risk identification:** In risk identification we are finding out what may happen to threaten the process, The identified risks are evaluated in the risk analysis and ordered relatively to each other
- Useful techniques are:
  - Lessons learned
  - Checklists
  - Risk workshops
  - Brainstorms
  - Expert interviews
  - Independent risk assessments
- » Risk mitigation
- » Risk follow-up
- **Risk analysis:** Risk analysis is the study of the identified risks. The analysis must be performed by all the appropriate stakeholders
- A **risk template** or a risk register is a very useful tool in risk management
- A risk template should include:
  - Risk identification (e.g., number or title)
  - Risk description
  - Probability
  - Effect
  - Exposure

- Test priority
- Mitigation action = test type
- Dependencies and assumptions

- **Perception of Risks:** In fact most risk analysis is based on perceptions; it is usually not possible to determine risk probability and effect totally objectively.
- The descriptions encompass:
  - » Project managers
  - » Developers
  - » Testers
  - » End users
- Scales for Risk Analysis
  - We can work with two different kinds of scale, namely:
    - » Qualitative
    - » Quantitative
- **Effect:** The effect is the impact or consequences of a risk if (when?) it occurs. Quantitative scale for the effect is the actual cost imparted by a failure occurring out in the field. The actual cost can be measured in any agreed currency
- Final effect = Σ(effect*weight) / Σ(weight)
- **Probability:** The probability is the likelihood of the materialization of a risk. We first of all need to agree on a scale.
- On a quantitative scale probability can be measured on a scale from 0 to 1 or a scale from 0% to 100%.
- **Risk level :** calculated for each of the identified risks as
- Risk level = final effect x final probability
- The distribution of the final risk level over individual risks is used to plan the test activities.
- It can be used to prioritize the test activities and to distribute the available time and other resources according to the relative risk level.

Risk Mitigation:

- We use the results of the risk analysis as the basis for the risk mitigation.
- "To mitigate" means "to make or become milder, less severe or less painful.
- Faced with the list of risks and their individual risk level we have to go through each of the risks and decide:
  - What we are going to do
  - How we are going to do it
  - When we are going to do it

III-BCA

UNIT-4

TEST TECHNIQUES

Test case design techniques are the heart of testing. Test techniques support systematic and meticulous work and make the testing specification effective and efficient.

1. SPECIFICATION-BASED TECHNIQUES

→ These techniques are used to design test cases based on an analysis of the description of the product without reference to its internal workings.

→ These techniques are also known as black-box tests.

→ The following are specification based testing techniques.

① Equivalence partitioning and boundary value analysis
② Decision Tables
③ Cause-Effect graph
④ state transition testing
⑤ classification Tree method
⑥ pair wise testing
⑦ use case testing

① EQUIVALENCE PARTITIONING & BOUNDARY VALUE ANALYSIS

→ In this we partition the input & output domain into equivalence classes.

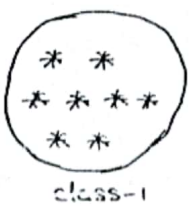
→ All the members of the class belongs to only one class.

→ No member belongs to more than one class.

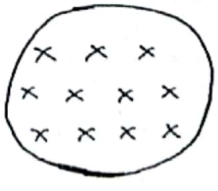
→ A member must belongs to a class which couldn't be out side of any class.

→ When we partition a domain into equivalent classes we may get both valid classes and invalid classes.

Elements



class-1



class-2.

Boundary value Analysis

→ Boundary value analysis is/are related to equivalence class partitioning.

→ Boundary value analysis is the process of identifying the boundary values.

→ A boundary value is the value on a boundary of an equivalence class.

→ It is not difficult to identify the boundary values for interval classes with precise boundary.

→ If a class has imprecise boundary, the boundary value is one increment inside the imprecise boundary.

→ Consider the interval

$$0 <= \text{income} <= 500.$$

For this equivalence class the boundaries are 0, 500.

→ consider the interval

$$0 <= \text{income} < 500.$$

For this equivalence class the boundaries are 0, 499.

→ As we have valid and invalid equivalence classes, we also have valid and invalid boundary values.

② DECISION TABLES

A decision table is a table which contains the actions of the system depending on certain combinations of input conditions.

→ Decision Tables are used to express rules and regulations for embedded systems and administrative systems.

→ Decision Tables are useful to provide an overview of combinations of inputs and the resulting output.

Decision Tables are brilliant for determining the complete requirements.

→ When there are 'n' number of input conditions, decision tables always have 2^n columns.

DECISION TABLE TEMPLATE

	TC1	TC2	---	TCn.
input condition-1				
input condition-m				
action-1				
action-m				

③ CAUSE - EFFECT GRAPH

→ A cause-effect graph is a graphical way of representing inputs with their associated outputs.

→ Here inputs are called causes and outputs are called effects.

→ This technique is used to design test cases for functions that depend on a combination of more input items.

→ Any functional requirement can be expressed as

f (old state, input) → (new state, output)

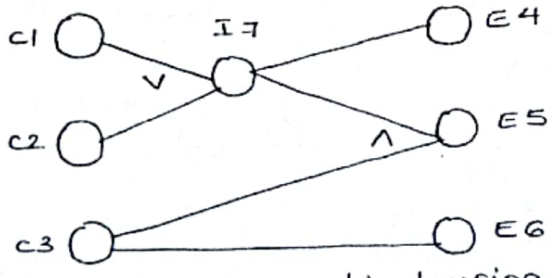
A cause-effect graph is constructed as

↳ list and assign an ID to all causes.

↳ list and assign an ID to all effects.

↳ for each effect make a boolean expression so that the effect is expressed in terms of relavent causes.

↳ draw the cause-effect graph.



∨ → causes are combined using OR

∧ → causes are combined using AND.

④ STATE TRANSITION TESTING

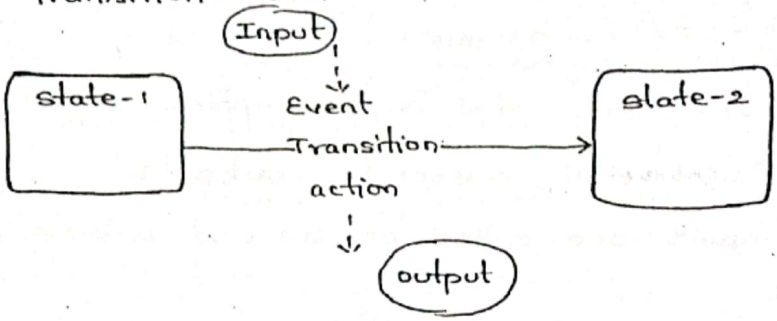
→ state transition testing is based on a state machine model of the test object.

→ It is a design technique used for embedded software, also applicable for user interface design.

→ A state is a collection of features of the system at a given point in time.

→ The transition from one state to another state is initiated by an event.

A transition = start state + event + action + end state

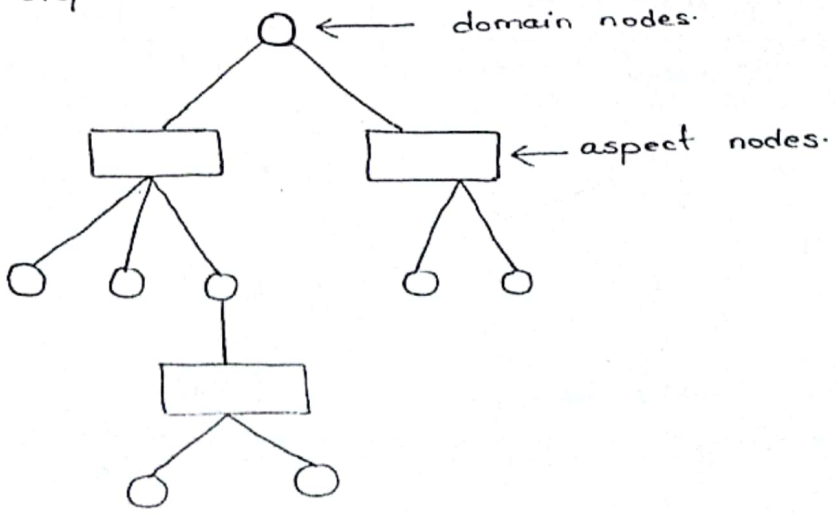


⑤ CLASSIFICATION TREE METHOD

→ The classification tree method is a way to partition input and state domains into classes.

→ This method can handle more complex situations where input and output domains looked from multiple point of views.

→ In the classification tree method we can partition a domain in several ways and refine each partition as step by step.



PAIRWISE TESTING

→ Pairwise testing is used to test products with all types of possible combinations of inputs.

→ Pairwise testing reduces the number of test cases compared to testing all combinations.

→ There are two techniques in pairwise testing.

1. Orthogonal arrays
2. allpairs algorithm.

⑦ USE CASE TESTING

→ A use case shows how the product interacts with one or more actors.

→ A use case includes a number of actions performed by the product as a results of events from actors.

→ An actor may be a user or other product.

→ Use cases are much used to express user requirements.

→ Testing a use case involves testing the main flow.

———— o ————

2. STRUCTURE BASED TECHNIQUES

→ The structural test case design techniques are used to design test cases based on an analysis of the internal structure of the component or system.

→ These techniques are also called as White Box tests.

→ These techniques focus on the testing of code and they are primarily used for component testing and low-level integration testing.

→ The following are structural test case design techniques.

① statement testing
② decision testing / branch testing
③ condition testing
④ multicondition testing
⑤ condition determination testing
⑥ loop testing
⑦ path testing.

① STATEMENT TESTING

→ Statement testing is a test case design technique in which, test cases are designed to execute statements.

→ A statement might be simple or compound.

→ For a simple statement there will be only one statement, for compound there will be multiple statements.

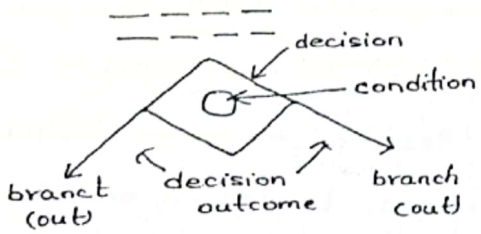
→
```
c = a+b ;        is one statement.

if (a==2) then
    c = b/a
endif;           is more than one statement.
```

→ In statement testing we design test cases to get a specifically required statement coverage.

→ Statement coverage is the percentage of executable statements in a test.

② DECISION / BRANCH TESTING

→ It is very difficult to define branches correctly, where as decisions and decision outcomes are very simple to define.



→ To define test cases for decision testing we have to

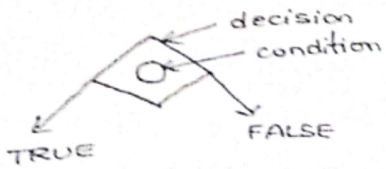
- Divide the code into basic blocks.
- Identify the decisions
- Design test cases to cover the decision outcomes or branches.

→ In most cases a decision has two outcomes, but it is possible for a decision to have more outcomes.

③ CONDITION TESTING

→ A condition is an expression that can be evaluated to be either TRUE or FALSE.

→ A statement which contains AND or OR is not a condition, because AND, OR are Boolean operators.



→ We will have 2 test cases for each condition.

④ MULTIPLE CONDITION TESTING

→ In multiple condition testing we test combinations of condition outcomes.

→ To get 100% coverage we have to test all combinations of outcomes of all conditions.

→ Because there are two possible outcomes for each condition it requires 2^n test cases where n is the number of conditions.

```
if (X OR (Y AND Z)) then
    - - -
    - - -.
```

Here we require 8 test cases.

Test case	X	Y	Z
1	TRUE	TRUE	TRUE
2	FALSE	TRUE	TRUE
3	FALSE	FALSE	TRUE
4	FALSE	FALSE	FALSE
5	FALSE	TRUE	FALSE
6	TRUE	FALSE	FALSE
7	TRUE	TRUE	FALSE
8	TRUE	FALSE	TRUE

⑤ CONDITION DETERMINATION TESTING

→ In this technique we design test cases to execute branch condition outcomes that independently affect a decision outcome.

→ Condition determination coverage depends on how the conditions are combined in the decision statements

↳ minimum we need n+1 test cases

↳ maximum we need 2^n test cases

where n → no. of conditions.

if (X OR (Y AND Z))

TEST CASE	X	Y	Z
1	TRUE	–	–
2	FALSE	TRUE	TRUE
3	FALSE	FALSE	–
4	FALSE	TRUE	FALSE

Here '–' indicates result of that condition outcome will not impact on the result.

⑥ LOOP TESTING

→ A loop is repetitive execution of the same statements with different variable values.

→ For some types of loops the loop body will always be executed at least once; for others it may be skipped depending on the condition of the looping.

→ To design test cases for this

↳ identify each code line by its number

↳ list the branches

↳ from the list, find start point of LCSAJ

↳ derive LCSAJ from each start point.

```
Read A;
B = 1;
While A <= 100 do
 B = B+1;
 A = A+1;
end While;
Write A;
Write B;
```

⑦ PATH TESTING

→ A path is a sequence of executable statements in a component from an entry point to an exit point.

→ The possible no. of paths through a component is exponentially linked with the no. of decisions, decisions involving loops.

——— o ———

TECHNIQUES

→ In this technique we are looking at the types of defects we might find in the product under testing.

TAXONOMIES

→ A taxonomy is an ordered hierarchy of names.

→ A taxonomy is a sort of checklist over defects to look for, and is used to design test cases.

1. Requirements.
   - 1.1 Requirements incorrect
   - 1.2 Requirements logic
   - 1.3 Requirements completeness
   - 1.4 Verifiability
   - 1.5 Presentation, documentation
   - 1.6 Requirement changes
2. Features and Functionalities
   - 2.1 Feature/Function correctness
   - 2.2 Feature completeness
   - 2.3 Functional case completeness
   - 2.4 Domain bugs
   - 2.5 User messages and diagnostics
   - 2.6 Exception conditions mishandled
3. Structural Bugs
   - 3.1 Control flow and sequencing
   - 3.2 processing
4. Data
   - 4.1 Data definition and structure
   - 4.2 Data access and handling
5. Implementation and coding
   - 5.1 Coding and typographical
   - 5.2 Style and standards violation
   - 5.3 Documentation
6. Integration
   - 6.1 Internal interfaces
   - 6.2 External interfaces, timing, throughput

7. System and Software Architecture
   - 7.1 O/S call and use
   - 7.2 Software architecture.
   - 7.3 Recovery and accountability
   - 7.4 Performance
   - 7.5 Incorrect diagnostics, exceptions
   - 7.6 Partitions, overlay
   - 7.7 Sysgen, environment
8. Test Definition and Execution
   - 8.1 Test design bugs
   - 8.2 Test execution bugs
   - 8.3 Test documentation
   - 8.4 Test case completeness.

STATIC ANALYSIS

STATIC ANALYSIS OF CODE

The static techniques for code are

① control flow analysis

② data flow analysis

③ compliance to standards

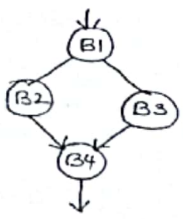
④ calculation of code metrics

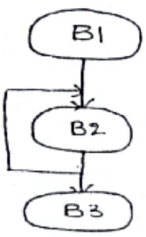
① Control flow analysis

→ A control flow is an abstract representation of all possible sequence of events in the execution.

→ Control flow goes through basis blocks or nodes of code being executed as an entity.

→ static analysis tools may be used to draw control flow graphs.





Data flow analysis

→ Data flow specifies the life cycle for a data item, a variable, consists of phases:

↳ declaration – Space is reserved in memory for the variable value.

↳ definition – A value is assigned to the variable.

↳ use – The value of the variable is used

↳ Destruction – The memory set aside for the value of the variable is freed for others to use.

→ static analysis tools can find anomalies in the data flow in relation to the variable life cycle.

③ Compliance to code standard

→ A coding standard is a guideline for the layout of the code being written in an organization.

→ static analysis tools can check for standard coding style violations.

Advantages:

↳ fewer faults in code

↳ easier component testing

↳ easier maintenance.

④ Calculation of code Metrics

→ static analysis tools provide measurements of different aspects of the code like:

↳ size measures – no. of lines of code, no. of comments

↳ complexity

↳ number of nested levels

↳ number of function calls.

———— ∪ ————

UNIT V
TESTING OF SOFTWARE CHARACTERISTICS

INTRODUCTION:

- A standard definition of the quality of a product is the degree to which the product fulfills the expectations the customers and users have for it.
- Quality attributes represent a way of structuring and expressing the expectations for a product.
- Testers to understand what quality attributes are and how they may be expressed.
- The ISO 9126 standard, a standard providing a quality model for product quality attributes.
  - Functionality
  - Reliability
  - Usability
  - Efficiency
  - Maintainability
  - Portability
- The test analyst is concerned with the functional and the usability quality attributes.
- The technical test analyst is concerned with the other four quality attributes

QUALITY ATTRIBUTES FOR TEST ANALYSTS:

- The functionality is what the product can do. Without functionality we don't have a product at all.
- Functionality may be tested at almost all test levels
- ISO 9126 breaks the functionality attribute into:
  - Suitability
  - Accuracy
  - Interoperability
  - Security

Functional Testing:

a) Suitability Testing:

- In suitability testing we test the requirements or needs concerned with the presence and appropriateness of a set of functions for specified tasks and user objectives
- Suitability of a product is about how the functionality of the software product supports the tasks the users are performing
- what detailed suitability could concern is:
  - Data availability (i.e., how do we get data, both in terms of background or reference data and/or data from other system information and in terms of data input and change

facilities and the associated levels of data validations)

- Data handling (i.e., what is data used for, for example, as event-driven interrupts or signals and—not least: calculations, actions, and so forth based on data and other input)
- Result presentation (i.e., output facilities, for example, in terms of windows and reports)

- Suitability testing can take place at all testing levels.

b) Accuracy Testing :

- In accuracy testing we test the requirements or needs concerned with the product's ability to provide the right or agreed upon results or effects.
- A more detailed accuracy specification could concern:
  - Algorithmic accuracy: Calculation of a value from other values and the correctness of function representation
  - Calculation precision: Precision of calculated values
  - Time accuracy: Accuracy of time related functionality
  - Time precision: Precision of time related functionality
- If in doubt about these attributes during testing it is better to ask, rather than to assume, especially if you are not an absolute domain expert.
- Accuracy can be tested at all testing levels, the earlier the better.

c) Interoperability Testing

- In interoperability testing we test the requirements or needs concerned with the ability of our software system to interact with other specified systems.
- No software system stands alone; it will always have to interact with other systems in the intended deployment environment, such as hardware, other software systems like operating systems, database systems, browsers, and be-spoken systems, external data repositories, and network facilities.
- Detailed interoperability could concern:
  - Inbound interoperability: Ability to use output from standard, third party, or in-house products as input
  - Outbound interoperability: Ability to produce output in the format used by standard, third-party, or in-house products
  - Spawnability: Ability to activate other products
  - Activatability: Ability to be activated by other products
- Interoperability testing usually takes place at the system integration testing level.

d) Functional Security Testing

- In security testing we test the requirements or needs concerned with the ability to prevent unintended access and resist deliberate attacks intended to gain unauthorized access to confidential information, or to make unauthorized modifications

- Detailed security could be concerning:
  - Activity auditability: Log facilities for activities, actors, and so forth
  - Accessability: Access control mechanisms
  - Self-protectiveness: Ability to resist deliberate attempts to violate access control mechanisms
  - Confinement: Ability to avoid accidental unauthorized access to facilities outside the application
  - Protectiveness: Ability to resist deliberate attempts to access unauthorized facilities outside the application
  - Data integrity: Protection against deliberate damage of data
  - Data privacy: Protection of data against unauthorized access
- Security testing can be split into functional security testing and technical security testing.

Usability Testing:

- Usability is the suitability of the software for its users, in terms of the effectiveness, efficiency, and satisfaction with which specified users can achieve specified goals in particular environments or contexts of use.
- The effectiveness of a software product is its capability to enable users to achieve specified goals with accuracy and completeness.
- The efficiency of a product is its capability to enable users to expend appropriate amounts of resources in relation to the effectiveness achieved.
- Satisfaction of a product is its capability to satisfy users.

Users Concerned with Usability

- The usability attribute is related to users.
- A user group for a product is a group of people who will be affected in similar ways by the product.
- This group may indeed be divided into frequent users, occasional users, and rare users, or other relevant subgroups
- Age (e.g., preschool, children, teens, young adults, mature adults, and elderly)
- Attitude (e.g., hostile, neutral, enthusiastic)
- Cultural background
- Education (e.g., no education yet, illiterate, basic education, middle
- education, workman, university education)
- Disabilities (e.g., people who are dyslexic, color-blind, blind, partially sighted, deaf, mobility- impaired, or cognitively disabled)
- Gender
- Intelligence

Usability Sub-attributes

- ISO 9126 breaks the usability attribute into the following sub-attributes:
  - Understandability
  - Learnability
  - Operability
  - Attractiveness
- **Understandability has got to do with how difficult it is to recognize the logical concept and find out how to apply it in practice**
  - Extent to which the system maps the concepts employed in the business procedures
  - Extent to which existing nomenclature is used
  - Nature and presentation of structure of entities to work with
  - Presentation of connections between entities
- **Learnability concerns the learning curve for the product**
  - Extent to which a user of the system can learn how to use the system without external instruction
  - Presence and nature of on-line help facilities for specified parts of the system
  - Presence and nature of off-line help facilities for specified parts of the system
  - Presence and nature of specific manuals
- **Operability is about what the product is like to use and control in deployment.**
  - Presence and nature of facilities for interactions with the product
  - Consistency of the man-machine interface
  - Presence, nature, and ordering of elements on each form
  - Presence and nature of input and output formats
  - Presence and nature of means of corrections of input
  - Presence and nature of navigational means
  - Number of operations and/or forms needed to perform a specified task
  - Format, contents, and presentation of warnings and error messages
  - Presence and nature of informative messages
  - Pattern of human operational errors over stated periods of time under
  - stated operational profiles according to defined reliability models

- **Attractiveness has got to do with how the users like the system and what may make them choose to acquire it in the first place. This may cover the:**
  - Use of colors
  - Use of fonts
  - Use of design elements, such as drawings and pictures
  - Use of music and sounds
  - Use of voices (male and female), languages, and accents
  - Layout of user interfaces and reports
  - Presence and nature of nontechnical documentation material
  - Presence and nature of technical documentation material
  - Presence and nature of specified demonstration facilities
  - Presence and nature of marketing material

Accessibility

- In recent years there has been more and more focus on equal opportunities, not the least of which are for people with disabilities.
- This also concerns software systems, which must be accessible and operable for everybody
- Accessibility may cover the:
  - Use of colors, especially mixtures of red and green
  - Possibility of connecting special facilities, such as speaker reading the text aloud, Braille keyboard, voice recognition, and touch screens
  - Possibility of using the product entirely by key strokes and/or voice commands
  - Facilities for multiple key pressure using only one finger or other pointing device
  - Possibility of enlarging forms and/or fonts
  - Navigation consistency

Establishing Usability Requirements

- Usability requirements should be expressed as explicitly as possible.
- Usability requirements can be derived from usability assessments
- Usability assessment is a requirements elicitation technique—and it should be performed early.
- A usability assessment is performed by representative users who are given tasks to complete ona prototype of the products.
- This can be hand-drawn
- Sketches of forms or mock-ups of the forms made in, for example, power point.

- Any thoughts and difficulties the users have in completing the tasks are recorded.
- After the usability assessment the comments are analyzed; the prototype may be changed and assessed again; and finally the usability requirements are derived.
- Usability assessments can be done very primitively by review of prototypes and storyboards

Testing Usability

- Usability may be tested in various ways during the development life cycle.
- Techniques to use may be:
  - Static tests
  - Verification and validation of the implementation
  - Surveys and questionnaires
  - Static tests can be performed as reviews and inspections of usability specifications.
  - The verification and validation of the implementation of the usability requirements are performed on the working system.
  - Surveys and questionnaires may be used where subjective measures, such as the percentage of representative future users who like or dislike the user interface, are needed.

QUALITY ATTRIBUTES FOR TECHNICAL TEST ANALYSTS:

- The functionality will always behave and present itself in certain ways.
- we call the non-functional or functionality- sustaining attributes

Some classics standards, which have been around for quite some time, are listed here with the quality attributes they include:

ISO 9126	Functionality, Reliability, Usability, Efficiency, Maintainability, Portability
McCall and Matsumoto	Integrity, Correctness, Reliability, Usability, Efficiency, Maintainability, Testability, Flexibility, Portability, Interoperability, Reusability
IEEE 830	Performance, Reliability, Availability, Security, Maintainability, Portability
ESA PSS-05	Performance, Documentation, Quality, Safety, Reliability, Maintainability

- The nonfunctional requirement types covered in this section are:
  - Reliability
  - Efficiency
  - Maintainability
  - Portability

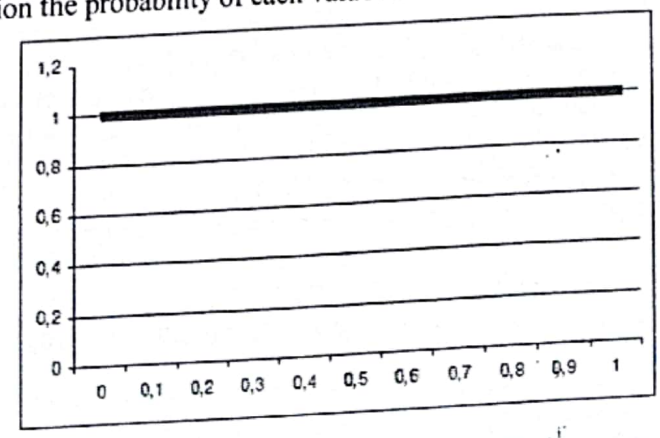
Technical Testing in General:

- In principle the non-functional testing is identical to the functional testing; it should be based on requirements and needs and use the test case design techniques.
- Testers should, help developers and analysts define these requirements from the beginning; and we can review the requirements to ensure that they are comprehensive and testable.
- It is important that the non-functional requirements are measurable and testable.
- The circumstances under which the goal is to be achieved must also be specified.
- It must be remembered that each non-functional requirement must be expressed using a scale, a specific goal, and possibly also acceptable limits.
- The testing of the nonfunctional quality attributes must be executed in a realistic environment reflecting the specified circumstances.
- This can be in terms operational profiles.
- An operational profile is a description of:
    - How many
    - Of which user groups
    - Will use what parts of the system
    - When
    - How much and/or how often
- If care is not taken to ensure realistic circumstances the testing may be a complete waste of time and, even worse, create a false sense of confidence in the product.

Random Input Technique:

- The random input technique can assist in generating input data based on a model of the input domain that defines all possible input values and their operational distribution.
- Random input follows the input distribution; the input values are constrained and guided by this.
- Expected input patterns can be estimated or may be known before deployment, and that knowledge can be used in testing.

In uniform distribution the probability of each value in the value domain is equal.



- For random testing we select input values for the test cases randomly from the input domain according to the input distribution.
- If the distribution is unknown, we can always use a uniform distribution.
- Random input is mainly interesting when the objective is to get the system to crash.
- A large number of test cases can be generated quite quickly, and long sequences of input can be run.
- The expected result and the actual result are, however, usually not that important when we are performing reliability or performance testing.
- The benefit of random input is that it is very cost-effective, especially if automated
- Another benefit is that random input testing may find "unexpected" combinations and sequences and may detect initialization problems.

Technical Security Testing:

- In security testing we test the requirements or needs concerned with the ability to prevent unintended access and resist deliberate attacks intended to gain unauthorized access to confidential information.
- Detailed security attributes may be:
  - Activity auditability: Log facilities for activities, actors, and so on
  - Accessability: Access control mechanisms
  - Self-protectiveness: Ability to resist deliberate attempts to violate
  - access control mechanisms
  - Confinement: Ability to avoid accidental unauthorized access to
  - facilities outside the application
  - Protectiveness: Ability to resist deliberate attempts to access
  - unauthorized facilities outside the application
  - Data integrity: Protection against deliberate damage of data
  - Data privacy: Protection of data against unauthorized access
- Security testing can be split into functional security testing and technical security testing

Reliability Testing:

Reliability is the probability that software will not cause the failure of a system for a specified time under specified conditions.

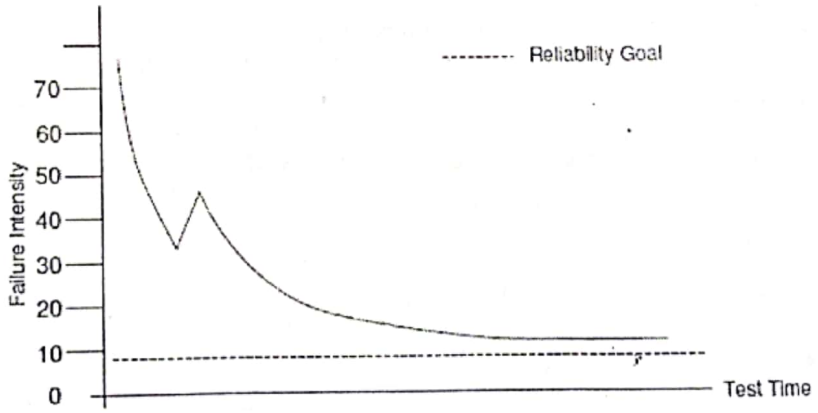
- Concerning reliability we have got to be realistic: It is impossible to produce 100% fault-free products!
- Functional testing and reliability testing are connected. The goal of functional testing is to obtain the highest possible reliability of the product within the given limits.
- The reliability testing should be based on operational profiles specified for the product,

and reliability goals expressed in requirements.

- A specific reliability may be used as a test completion or test exit criterion for the system testing, allowing for earlier reliability testing
- The ISO 9126 standard breaks the reliability attributes down into a number of sub attributes. There are:
  - Maturity
  - Fault tolerance (robustness)
  - Recoverability

Maturity Testing:

- Maturity is the frequency of failures as a result of faults in the software.
- A product's expected maturity is often expressed in terms of
  - Mean time between failures (MTBF)
  - Failures per test hour
  - Failures per production time (typically months)
  - Failures per number of transactions
  - The results of the reliability testing can be graphic presentations of the measurements using a reliability growth model



Reliability Growth Curve Reflecting Out-of Control Process

Robustness Testing:

- Fault tolerance or robustness is the product's ability to maintain a specified level of performance in the presence of software defects or infringement of a specified interface.
- This may cover the product's:
  - Containment of defects to specified parts of the system
  - Reactions to failures of a given severity
  - Self-monitoring of the operations and self-identification of defects Ability to allow specified work to continue after a failure of specified
  - Severity for specified parts of the system under specified conditions
  - Loss of specified operations in case of failure of specified severities in specified periods of

time for specified parts of the system

- Loss of specified data in case of failure of specified severities in specified periods of time for specified parts of the system

Recoverability Testing:

- Recoverability is the product's ability to reestablish its required level of performance and recover the data directly affected after a failure.
- This may cover aspects like:
  - Downtime after a failure of specified severities in specified periods of time for specified parts of the system
  - Uptime during specified periods of time for specified parts of the system over a specified period of time
  - Downtime during specified periods of time for specified parts of the system over a specified period of time
  - Time to reestablish consistent data in case of failure causing inconsistent data
  - Built-in backup facilities
  - Need for duplication (standby machine)
  - Redundancy in the software system
  - Reporting of effects of a crash
  - "Advice" in connection with restart

Efficiency Testing:

- In efficiency testing we test the requirements or needs concerned with the product's ability to provide appropriate performance, relative to the amount of resources used, under stated conditions.
- The ISO 9126 standard breaks the efficiency attributes down into the subattributes:
  - Time behavior (performance)
  - Resource utilization

Performance Testing: Time behavior or performance consists of the expectations towards the product's ability to provide appropriate response and processing time and throughput rates when performing its functions under stated conditions

Load Testing: Load testing is a special subtype of performance testing concerned with the product's behavior under specified load conditions.

Stress Testing: Stress is an expression of the product's capability for handling extreme situations

Scalability Testing: Scalability is the product's ability to meet future efficiency requirements.

Maintainability Testing:

- In maintainability testing we test the requirements or needs concerned with the product's ability to be analyzed and modified.
- Modifications may include corrections of defects, improvements or adaptations of the software to changes in the environment, and enhancements in requirements and functional specifications.
- The ISO 9126 standard breaks the efficiency attributes down into the subattributes:
  - Analyzability
  - Changeability
  - Stability
  - Testability
- Maintainability testing can be performed as static testing, where the structure, complexity, and other attributes of the code and the documentation are reviewed or undergoing inspections based on the pertaining maintenance requirements

Analyzability Testing:

- Analyzability is the ability of maintainers to identify deficiencies, diagnose the cause of failures, and identify areas requiring modification to implement required changes.
- Analyzability may cover aspects like:
  - Understandability: Making the design documentation, including the source code, understood by maintainers
  - Design standard compliance: Adherence to defined design standards
  - Coding standard compliance: Adherence to defined coding standards
  - Diagnosability: Presence and nature of diagnostic functions in the code
  - Traceability: Presence of traces between elements, for example, between requirements and test cases, and requirements and design and code
  - Technical manual helpfulness: Nature of any technical manual or specification

Changeability Testing:

- Changeability is the capability for implementation of a specified modification in the product
- Changeability may cover aspects like:
  - Modularity: The structure of the software
  - Code change efficiency: Capability for implementing required changes
  - Documentation change efficiency: Capability for documenting
  - implemented changes

Stability Testing:

- Stability is the capability of the product to avoid unexpected effects from modifications of the software, that is, to the risk of unexpected effects from modifications.
    - Data cohesion: Usage of data structures
    - Refailure rate: Pattern of new failures introduced as an effect of
    - implementation of required changes

Testability Testing

- Testability is the capability of validating the modified system, that is, how easy it is to perform testing of changes, either new tests or confirmation test, and how easy it is to perform regression testing.

Portability Testing:

- In portability testing we test the requirements or needs concerned with the product's ability to be transferred into its intended environment.
- The environment may include the organization in which the product is used and the hardware, software, and network environment.
- The ISO 9126 standard breaks the portability attributes down into the subattributes:
    - Installability
    - Coexistence
    - Adaptability
    - Replaceability

Installability Testing:

- Installability is the capability of installing the product in a specified environment
    - Space demand: Temporary space to be used during installation of the software in a specified environment.
    - Checking prerequisites: Facilities to ensure that the target environment is meeting the demands of the product,
    - Installation procedures: Existence and understandability of installation aids such as general or specific installation scripts, installation manuals, or wizards. This may also include requirements concerning the time and effort to be spent on the installation task.
    - Completeness: Facilities for checking that an installation is complete
    - Installation interruption: Possibility of interrupting an installation. and rolling any work done back to leave the environment unchanged
    - Customization: The capability of setting or changing parameters at
    - installation time in a specified environment.
    - Initialization: The capability of setting up initial information at installation
    - time, both internal and external in a specified environment.

- Deinstallation: Facilities for removing the product partly (downgrading)
- or completely from the environment.

Coexistence testing:

- Coexistence is the software product's capability to coexist with other independent software products in a common environment sharing common resources
- Coexistence can be very difficult to specify, since we don't always know into which environment our software product is being placed.
- Coexistence testing can also be very difficult to perform, since it is usually impossible to establish correct test environments for this.

Adaptability:

- Adaptability is the capability of the software product to be adapted to different specified environments without applying actions or means other than those provided for this purpose for the system

- Adaptability may cover aspects like:
  - Hardware dependency:
  - Software dependency:
- Representation dependency:
  - Standard language conformance:
  - Dependency encapsulation:
  - Text convertability:

Replaceability Testing:

- Replaceability is the capability of the product to be used in place of another specified product for the same purpose in the same environment.
- replaceability may cover aspects like
  - Data loadability
  - Data convertability

Shorts.

(1) Define & Explain about Sequential model?

(2) Explain about the testing Process

(3) Write the steps of master test plan

(4) Write some state transition testing template

(5) Explain about random input technique.

(6) Explain about the test closure

(7) Explain about the Data flow Analysis.

(8) Explain about the Efficiency tuting

Essays

1a) Explain about product paradigms.

(Or)

1b) Explain about matrix & measurements of Software Testing

2a) ~~Dife~~ Define testing planning. Explain about the testing planning and contiol

(Or)

2b) Explain the process of test implementation & Execution.

3a) Explain the overview of test Estimation

(Or)

3b) Define risk Explain about types of risks.

4a) Explain about the Structured Band techniques.

(Or)

4b) Define & Explain about taxonomies.

5a) Define & Explain about the usability testing.

(Or)

5b) Define reliability testing Explain about maturity testing.