Software Testing November 2016 Solution Set

(2½ hours)

Total Marks: 75

N. B.: (1) **All** questions are **compulsory**.
(2) Make**suitable assumptions** wherever necessary and **state the assumptions** made.
(3) Answers to the **same question** must be **written together**.
(4) Numbers to the **right** indicate **marks**.
(5) Draw **neat labeled diagrams** wherever **necessary**.
(6) Use of **Non-programmable** calculators is **allowed**.

1 Attempt *any two* of the following: **10**

a Should testing be done only after the code of the product is ready? Support your answer with a valid explanation.

Answer 1(a).

Testing activities should not start only after coding gets over.

Testing is a process rather than a single activity - there are a series of activities involved. All life cycle activities look at testing as a process that takes place throughout the software development life cycle. The later in the life cycle we find bugs, the more expensive they are to fix. If we can find and fix requirements defects at the requirements stage, that must make commercial sense. We'll build the right software, correctly and at a lower cost overall. So, the thought process of designing tests early in the life cycle can help to prevent defects from being introduced into code.

(If students have explained Testing throughout the SDLC using V-model, then also give marks)

b State and explain briefly software testing principles. (for any 5 – give full marks)

Answer 1(b).

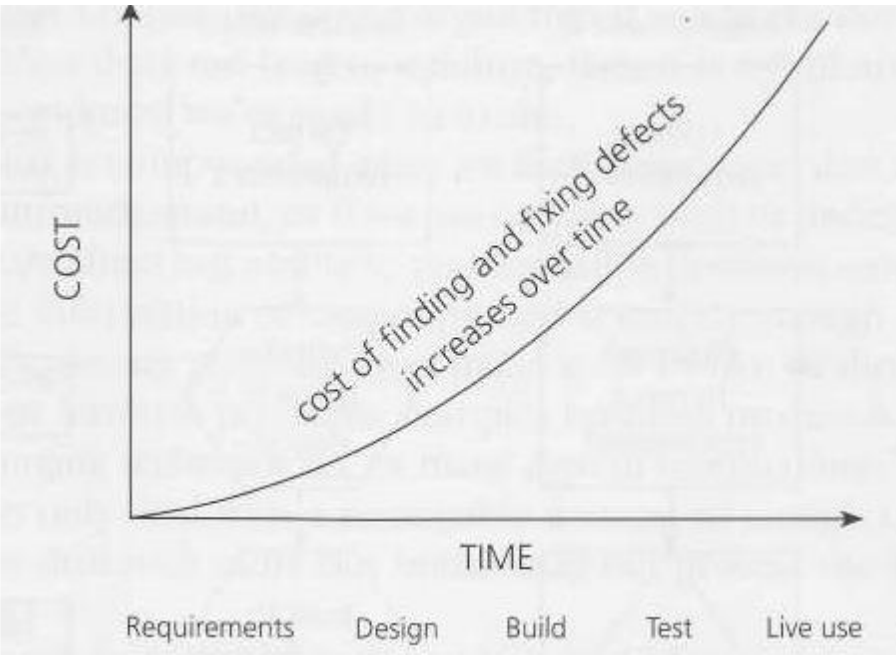
Principle 1:	**Testing shows presence of defects**	Testing can show that defects are present, but cannot prove that there are no defects. Testing reduces the probability of undiscovered defects remaining in the software but, even if no defects are found, it is not a proof of correctness.
Principle 2:	**Exhaustive testing is impossible**	Testing everything (all combinations of inputs and preconditions) is not feasible except for trivial cases. Instead of exhaustive testing, we use risks and priorities to focus testing efforts.
Principle 3:	**Early testing**	Testing activities should start as early as possible in the software or system development life cycle and should be focused on defined objectives.
Principle 4:	**Defect clustering**	A small number of modules contain most of the defects discovered during pre-release testing or show the most operational failures.
Principle 5:	**Pesticide paradox**	If the same tests are repeated over and over again, eventually the same set of test cases will no longer find any new bugs. To overcome this 'pesticide paradox', the test cases need to be regularly reviewed and revised, and new and different tests need to be written to exercise different parts of the software or system to potentially find more defects.
Principle 6:	**Testing is context dependent**	Testing is done differently in different contexts. For example, safety-critical software is tested differently from an e-commerce site.
Principle 7:	**Absence-of-errors fallacy**	Finding and fixing defects does not help if the system built is unusable and does not fulfill the users' needs and expectations.

c What is the difference between defect and failure in software testing? What is the cost of defect?

Answer 1(c).

If someone makes an **error** or mistake in using the software, this may lea directly to a problem - the software is used incorrectly and so does not behav as we expected. However, people also design and build the software and the can make mistakes during the design and build. These mistakes mean that ther are flaws in the software itself. These are called **defects** or sometimes bugs faults. Remember, the software is not just the code; check the definition of sof ware again to remind yourself.

When the software code has been built, it is executed and then any defects ma cause the system to fail to do what it should do (or do something it shouldn't causing a **failure.** Not all defects result in failures; some stay dormant in the cod and we may never notice them.



(2M-defect & Failure; 1-M for graph; 2M – for graph's explanation)

d Explain the activity of the "Fundamental Test Process" in which test environment is set up.
Answer 1(d).
During "Test implementation and execution", we set up the test environment.
(explanation of this activity)

2 Attempt *any two* of the following: **10**

a Give difference between black box and white box testing.
Answer 2(a).
The Differences Between Black Box Testing and White Box Testing are listed below.

Criteria	**Black Box Testing**	**White Box Testing**
Definition	Black Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is NOT known to the tester	White Box Testing is a software testing method in which the internal structure/ design/ implementation of the item being tested is known to the tester.
Levels Applicable To	Mainly applicable to higher levels of testing:Acceptance Testing System Testing	Mainly applicable to lower levels of testing:Unit Testing Integration Testing
Responsibility	Generally, independent Software Testers	Generally, Software Developers
Programming Knowledge	Not Required	Required

Implementation Knowledge	Not Required	Required
Basis for Test Cases	Requirement Specifications	Detail Design

(Give full marks for any 5 relevant points)

b Give difference between re-testing and regression testing.

Answer 2(b). (If various points of difference are not written but the core difference is written properly award 3-4 marks.)

Difference between Retesting and Regression Testing.

Regression testing	**Retesting**
Regression testing is done to find out the issues which may get introduced because of any change or modification in the application.	Retesting is done to confirm whether the failed t cases in the final execution are working fine or r after the issues have been fixed.
The purpose of regression testing is that any new change in the application should NOT introduce any new bug in existing functionality.	The purpose of retesting is to ensure that the particular **bug** or issue is resolved and the functionality is working as expected.
Verification of bugs are not included in the regression testing.	Verification of bugs are included in the retesting
Regression testing can be done in parallel with retesting.	Retesting is of high **priority** so it's done before regression testing.
In case of regression testing the testing style is generic	In case of retesting the testing is done in a plann way.
During regression testing even the passed test cases are executed.	During **retesting only failed test cases are re-executed**.
Regression testing is carried out to check for unexpected side effects.	Retesting is carried out to ensure that the origina issue is working as expected.

Regression testing is done only when any new feature is implemented or any modification or enhancement has been done to the code.	Retesting is executed in the same environment with same data but in new build.

c Which **testing level** tests interfaces between components and interactions to different parts of a system? Explain.

Answer 2(c)

Integration testing tests interfaces between components, interactions to different parts of a system such as an operating system, file system and hardware or interfaces between systems. (Explanation of Integration Testing should be written properly – Big Bang , Incremental(Top down, bottom up, Functional Incremental))

d Describe the role of regression testing and impact analysis within maintenance testing.

Answer 2(d)

Usually maintenance testing will consist of two parts:

• testing the changes

• regression tests to show that the rest of the system has not been affected by the maintenance work.

In addition to testing what has been changed, maintenance testing includes extensive regression testing to parts of the system that have not been changed.

A major and important activity within maintenance testing is impact analysis. During **impact analysis,** together with stakeholders, a decision is made on what parts of the system may be unintentionally affected and therefore need careful regression testing.

Risk analysis will help to decide where to focus regression testing - it is unlikely that the team will have time to repeat all the existing tests.

If the test specifications from the original development of the system are kept, one may be able to reuse them for regression testing and to adapt them for changes to the system. This may be as simple as changing the expected results for your existing tests.

Sometimes additional tests may need to be built. Extension or enhancement to the system may mean new areas have been specified and tests would be drawn up just as for the development. It is also possible that updates are needed to an automated test set, which is often used to support regression testing.

3 Attempt ***any two*** of the following: **10**

a How can we evaluate or analyse various documents like requirement document, design document, test plan or user manual etc.?

Answer 3(a)

One powerful technique that can be used to evaluate or analyse various documents like requirement document, design document, test plan or user manual etc.is “Static testing”, e.g. **reviews.** In principle all software work products can be tested using review techniques.

Types of defects that are easier to find during static testing are: deviations from standards, missing requirements, design defects, non-maintainable code and inconsistent interface specifications.

the use of static testing, e.g. reviews, on software work products has various advantages:

• Since static testing can start early in the life cycle, early feedback on quality issues can be established, e.g. an early validation of user requirements and not just late in the life cycle during acceptance testing.

• By detecting defects at an early stage, rework costs are most often relatively low and thus a relatively cheap improvement of the quality of software products can be achieved.
• Since rework effort is substantially reduced, development productivity figures are likely to increase.
• The evaluation by a team has the additional advantage that there is an exchange of information between the participants.
• Static tests contribute to an increased awareness of quality issues.
In conclusion, static testing is a very suitable method for improving the quality of software work products.

b Discuss briefly different types of reviews.
Answer 3(b)

1. Walkthrough
2. Technical Review
3. Inspection

(Very brief explanation of each one)

c Define metrics. Explain cyclomatic complexity metrics with help of an example.
Answer 3(c) (Metrics def. – 2M; Cyclomatic complexity – 2M; example – 1M)
Metric - quantitative measure of degree to which a system, component or process possesses a given attribute. "A handle or guess about a given attribute." – Number of errors found per person hours expended.

Cyclomatic Complexity is a software metric (measurement), used to indicate the complexity of a program. It is a quantitative measure of the number of linearly independent paths through a program's source code. It was developed by Thomas J. McCabe, Sr. in 1976.

Cyclomatic Complexity is computed using the control flow graph of the program: the nodes of the graph correspond to indivisible groups of commands of a program, and a directed edge connects two nodes if the second command might be executed immediately after the first command. Cyclomatic complexity may also be applied to individual functions, modules, methods or classes within a program.

One testing strategy, called basis path testing by McCabe who first proposed it, is to test each linearly independent path through the program; in this case, the number of test cases will equal the cyclomatic complexity of the program.

Mathematically, the cyclomatic complexity of a structured program[a] is defined with reference to the control flow graph of the program, a directed graph containing the basic blocks of the program, with an edge between two basic blocks if control may pass from the first to the second. The complexity M is then defined as[2]

$$M = E - N + 2P,$$

where

E = the number of edges of the graph.
N = the number of nodes of the graph.
P = the number of connected components.

(any example)

d Define exit criteria in software testing. What is its purpose? Cite few examples of exit criteria.
Answer 3(d)

Exit criterion is used to determine whether a given test activity has been completed or NOT. Exit criteria can be defined for all of the test activities right from planning, specification and execution.

Exit criterion should be part of test plan and decided in the planning stage.

As exhaustive testing is not possible exit criteria helps us in deciding when to stop testing.

Examples of Exit Criteria:

- Verify if All tests planned have been run.
- Verify if the level of requirement coverage has been met.
- Verify if there are NO Critical or high severity defects that are left outstanding.
- Verify if all high risk areas are completely tested.
- Verify if software development activities are completed within the projected cost.
- Verify if software development activities are completed within the projected timelines.

4 Attempt *any two* of the following: **10**

a Explain State Transition test design technique with help of an example.

Answer 4(a)

State transition testing is used where some aspect of the system can be described in what is called a 'finite state machine'. This simply means that the system can be in a (finite) number of different states, and the transitions from one state to another are determined by the rules of the 'machine'. This is the
model on which the system and the tests are based. Any system where you get a different output for the same input, depending on what has happened before, is a finite state system. A finite state system is often shown as a **state diagram.**

A state transition model has four basic parts:

• the states that the software may occupy
• the transitions from one state to another
• the events that cause a transition
• the actions that result from a transition

(any example – 2M)

b Define test basis and test condition. Discuss how to identify test conditions.

Answer 4(b)

Test basis is **defined** as the source of information or the document that is needed to write **test** cases and also for **test** analysis. **Test basis** should be well **defined** and adequately structured so that one can easily identify **test** conditions from which **test**cases can be derived.

Test condition. An item or event of a component or system that could be verified by one or more **test** cases, e.g. a function, transaction, feature, quality attribute, or structural element.

A **test condition** is simply something that we could test. E.g. If we have a requirements specification, the table of contents can be our initial list of test conditions.

IDENTIFYING TEST CONDITIONS

* When identifying test conditions, we want to 'throw the net wide' to identify as many as we

can, and then we will start being selective about which ones to take forward to develop in more detail and combine into test cases. We could call them '**test possibilities**'.

* As we cannot test everything, we have to select a **subset** of all possible tests. In practice the subset we select may be a very small subset and yet it has to have a high probability of finding most of the defects in a system. We need some intelligent thought processes to guide our selection; **test techniques** are such thought processes.
* Each technique provides a set of rules or guidelines for the tester to follow in identifying test conditions and test cases.
* The test conditions that are chosen will depend on the test strategy or detailed test approach.

c Explain requirement traceability. Discuss its importance.

Answer 4(c)

Test conditions should be able to be linked back to their sources in the test basis - this is called **traceability.**

Traceability can be either horizontal through all the test documentation for a given test level (e.g. system testing, from test conditions through test cases to test scripts) or vertical through the layers of development documentation (e.g. from requirements to components).

Importance of Traceability :-

• The requirements for a given function or feature have changed. Some of the fields now have different ranges that can be entered. Which tests were looking at those boundaries? They now need to be changed. How many tests will actually be affected by this change in the requirements? These questions
can be answered easily if the requirements can easily be traced to the tests.

• A set of tests that has run OK in the past has started to have serious problems. What functionality do these tests actually exercise? Traceability between the tests and the requirement being tested enables the functions or features affected to be identified more easily.

• Before delivering a new release, we want to know whether or not we have tested all of the specified requirements in the requirements specification. We have the list of the tests that have passed - was every requirement tested?

d Explain the testing technique which is used when there is no specification, or if the specification is inadequate or out of date.

Answer 4(d)

Experience-based techniques are used to complement specification-based and structure-based techniques, and are also used when there is no specification, or if the specification is inadequate or out of date. This may be the only type of technique used for low-risk systems, but this approach may be particularly useful under extreme time pressure - in fact this is one of the factors leading to exploratory testing.

(brief explanation of Experience based testing including – Error Guessing & Exploratory Testing)

5 Attempt ***any two*** of the following: **10**

a Write a short note on configuration management.

Answer 5(a)

- Configuration management is in part about determining clearly what the items are that make up the software or system. These items include source code, test scripts, third-party software, hardware, data and both development and test documentation.
- Configuration management is also about making sure that these items are managed

carefully, thoroughly and attentively throughout the entire project and product life cycle.

- Configuration management has a number of important implications for testing. For one thing, it allows the testers to manage their testware and test results using the same configuration management mechanisms, as if they were as valuable as the source code and documentation for the system itself - which of course they are.
- Configuration management supports the build process, which is essential for delivery of a test release into the test environment. Simply sending Zip archives by e-mail will not suffice, because there are too many opportunities for such archives to become polluted with undesirable contents or to harbor left-over previous versions of items. Especially in later phases of testing, it is critical to have a solid, reliable way of delivering test items that work and are the proper version.
- Configuration management allows us to map what is being tested to the underlying files and components that make it up. This is absolutely critical. For example, when we report defects, we need to report them *against* something, something which is **version controlled.** If it's not clear what we found the defect in, the programmers will have a very tough time of finding the defect in order to fix it.
- When testers receive an organized, version-controlled test release from a change-managed source code repository, it is accompanied by a test item trans-mittal report or release notes.
- It provides a useful guideline for what goes into such a report. Release notes are not always so formal and do not always contain all the information shown.

IEEE 829 STANDARD: TEST
ITEM TRANSMITTAL REPORT TEMPLATE

Transmittal report identifier
Transmitted items
Location
Status
Approvals

b Write a short note on test progress monitoring.

Answer 5(b)

Test monitoring can serve various purposes during the project, including the following:

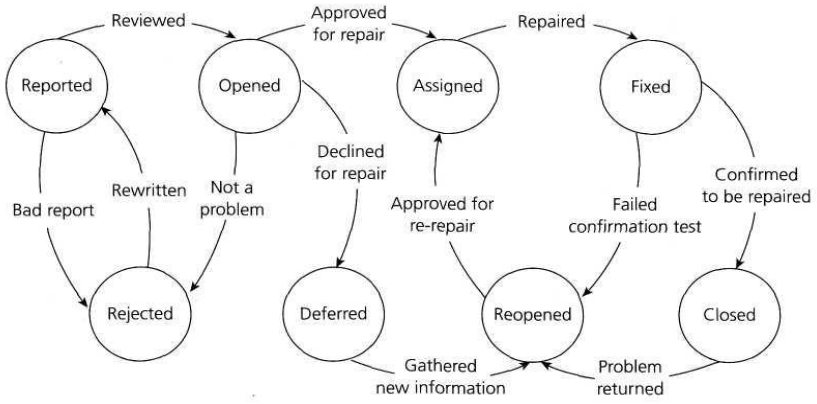
• Give the test team and the test manager feedback on how the testing work is going, allowing opportunities to guide and improve the testing and the project.
• Provide the project team with visibility about the test results.
• Measure the status of the testing, test coverage and test items against the exit criteria to determine whether the test work is done.
• Gather data for use in estimating future test efforts.

The following should be discussed briefly:-

1) IEEE 829 test log template.
2) Test Case Summary
3) Total defects opened and closed chart

c Explain Incident report life cycle with help of an example.

Answer 5(c) (explanation with diagram)



d For any risk, product or project, discuss four typical risk management options.

Answer 5(d)

For any risk, product or project, you have four typical options:

• Mitigate: Take steps in advance to reduce the likelihood (and possibly the impact) of the risk.

• Contingency: Have a plan in place to reduce the impact should the risk become an outcome.

• Transfer: Convince some other member of the team or project stakeholder to reduce the likelihood or accept the impact of the risk.

• Ignore: Do nothing about the risk, which is usually a smart option only when there's little that can be done or when the likelihood and impact are low.

6 Attempt *any two* of the following: **10**

a State the goals of a proof-of-concept or piloting phase for tool evaluation.

Answer 6(a)

The goals for a pilot project for a new tool are:

• to learn more about the tool (more detail, more depth);

• to see how the tool would fit with existing processes or documentation, how those would need to change to work well with the tool and how to use the tool to streamline existing processes;

• to decide on standard ways of using the tool that will work for all potential users (e.g. naming conventions, creation of libraries, defining modularity, where different elements will be stored, how they and the tool itself will be maintained);

• to evaluate the pilot project against its objectives (have the benefits been achieved at reasonable cost?).

b Discuss features of Test design tools.

Answer 6(b)

Test design tools help to construct test cases, or at least test inputs (which is part of a test case). If an automated oracle is available, then the tool can also construct the expected result, so it can actually generate test cases (rather than just test inputs).

Features or characteristics of test design tools include support for:

• generating test input values from:

- requirements;
- design models (state, data or object);
- code;
- graphical user interfaces;
- test conditions;

• generating expected results, if an oracle is available to the tool.

The benefit of this type of tool is that it can easily and quickly identify the tests (or test inputs) that

will exercise all of elements, e.g. input fields, buttons, branches. This helps the testing to be more thorough (if that is an objective of the test!)

c Discuss the benefits of using tools for testing.

Answer 6(c)

There are many benefits that can be gained by using tools to support testing, whatever the specific type of tool. Benefits include:

• reduction of repetitive work;
• greater consistency and repeatability;
• objective assessment;
• ease of access to information about tests or testing.

(brief explanation of each point is expected)

d What is "Test Comparator"? Discuss its importance and features.

Answer 6(d)

The essence of testing is to check whether the software produces the correct result, and to do that, we
must compare what the software produces to what it should produce.
A **test comparator** helps to automate aspects of that comparison.
There are two ways in which actual results of a test can be compared to the
expected results for the test.
1). Dynamic comparison is where the comparison is done dynamically, i.e. while the test is executing.
2).The other way is post-execution comparison, where the comparison is performed after the test has finished executing and the software under test is no longer running.

Features or characteristics of test comparators include support for:
• dynamic comparison of transient events that occur during test execution;
• post-execution comparison of stored data, e.g. in files or databases;
• masking or filtering of subsets of actual and expected results.

7 Attempt ***any three*** of the following: **15**

a Discuss the major tasks of Test Planning.

Answer 7(a)

- Determine the scope and risks and identify the objectives of testing
- Determine the **test approach** (techniques, test items, **coverage,** identifying and interfacing with the teams involved in testing, testware will be related to the requirements of the test strategy.
- Implement the test policy and/or the test strategy:
- Determine the required test resources (e.g. people, test environment, PCs).
- Schedule test analysis and design tasks, test implementation, execution and evaluation.
- Determine the **exit criteria.**

(1-2 lines explanation of each point)

b How do we check that we are building the system right? Explain.

Answer 7(b)

Explain "Verification".

Criteria	Verification
Definition	The process of evaluating work-products (not the actual final product) of a development phase to determine whether they meet the specified requirements for that phase.
Objective	To ensure that the product is being built according to the requirements and design specifications. In other words, to ensure that work products meet their specified requirements.
Question	Are we building the product *right*?
Evaluation Items	Plans, Requirement Specs, Design Specs, Code, Test Cases
Activities	• Reviews • Walkthroughs • Inspections

c Discuss briefly the features of "Static code analysis tools".

Answer 7(c)

- Coding standard
- Code Metrics
- Code Structure

  • control flow structure;
  • data flow structure;
  • data structure.

d Postal rates for 'light letters' are 25p up to l0g, 35p up to 50g plus an extra l0p for each additional 25g up to l00g. Design test case for weight of the letters using equivalence partitioning.

e Discuss the fundamental techniques of estimation for testing.

Answer 7(e)

We could start with a work-breakdown structure that identifies the stages, activities and tasks.

Starting at the highest level, we can break down a testing project into **phases** using the fundamental test process: planning and control; analysis and design; implementation and execution; evaluating exit criteria and reporting; and test closure.

Within each phase we identify **activit**ies and within each activity we **identify tasks** and perhaps subtasks.

To identify the activities and tasks, we work both **forward and backward.**

When we say we work forward, we mean that we start with the planning activities and then move

forward in time step by step, asking, 'Now, what comes next?'
Working backward means that we consider the risks that we identified during risk analysis.

f. Define test scripts. Explain briefly advanced scripting techniques for test execution tools.

Answer 7(f)

Test script is used to describe the instructions to a test execution tool. An automation script is written in a programming language that the tool can interpret.

There are different levels of scripting. Five are described here :-

• linear scripts (which could be created manually or captured by recording a manual test);
• structured scripts (using selection and iteration programming structures);
• shared scripts (where a script can be called by other scripts so can be re-used - shared scripts also require a formal script library under configuration management);
• **data-driven scripts** (where test data is in a file or spreadsheet to be read by a control script);
• **keyword-driven scripts** (where all of the information about the test is stored in a file or spreadsheet, with a number of control scripts that implement the tests described in the file).