CURRICULUM REVISION PROJECT

2012

TEACHER GUIDE FOR

(Software Testing – 17624)

SIXTH SEMESTER
COMPUTER ENGINEERING GROUP

DEC 2014

MAHARASHTRA STATE

BOARD OF TECHNICAL EDUCATION, Mumbai

(Autonomous) (ISO 9001:2008) (ISO/IEC 27001:2005)

INDEX

1. 0 APPROACH TO CURRICULUM DESIGN

1.1 Background:

MSBTE is introducing the revised curriculum under 'G' scheme from the academic year 2012-13.

There are many institutions in the state running different diploma courses. In order to ensure uniform and effective implementation of the curriculum it is necessary that every teacher is aware of approach for curriculum design, educational principles to be adopted, learning resources to be used and evaluation methods. The teacher guide prepared for each subject will provide the inputs related to above mentioned aspects to achieve uniform and effective implementation of curriculum of various subjects.

1.2 CURRICULUM PHILOSOPHY

MSBTE has adopted systems approach while designing the scientific based curriculum since 1995. The same approach has been adopted while revising the curriculum in semester pattern.

Fig. No. 1 shows the systems diagram. This diagram provides the holistic view for curriculum designing, development, implementation and evaluation

The input to polytechnic education system is the students having 10+ qualifications. The teaching learning process occurs in the institution for six/eight semesters. The output of the system i. e. Diploma pass out is normally the input to industries. (Some students do go for higher education). While designing the curriculum the expectations of the industries play a major role. Due to globalization and competition the industries expect that pass outs have generic and technological skills along with right attitude.

To fulfill the needs derived from systems approach following conceptual framework is considered:

1.3 Curriculum:

"Curriculum is an educational program designed and implemented to achieve specified educational objectives"

This definition takes into account the fact that

- Education is purposeful
- There is an organized plan of action contemplated
- Such a plan is translated into action through appropriate strategies of implementation.

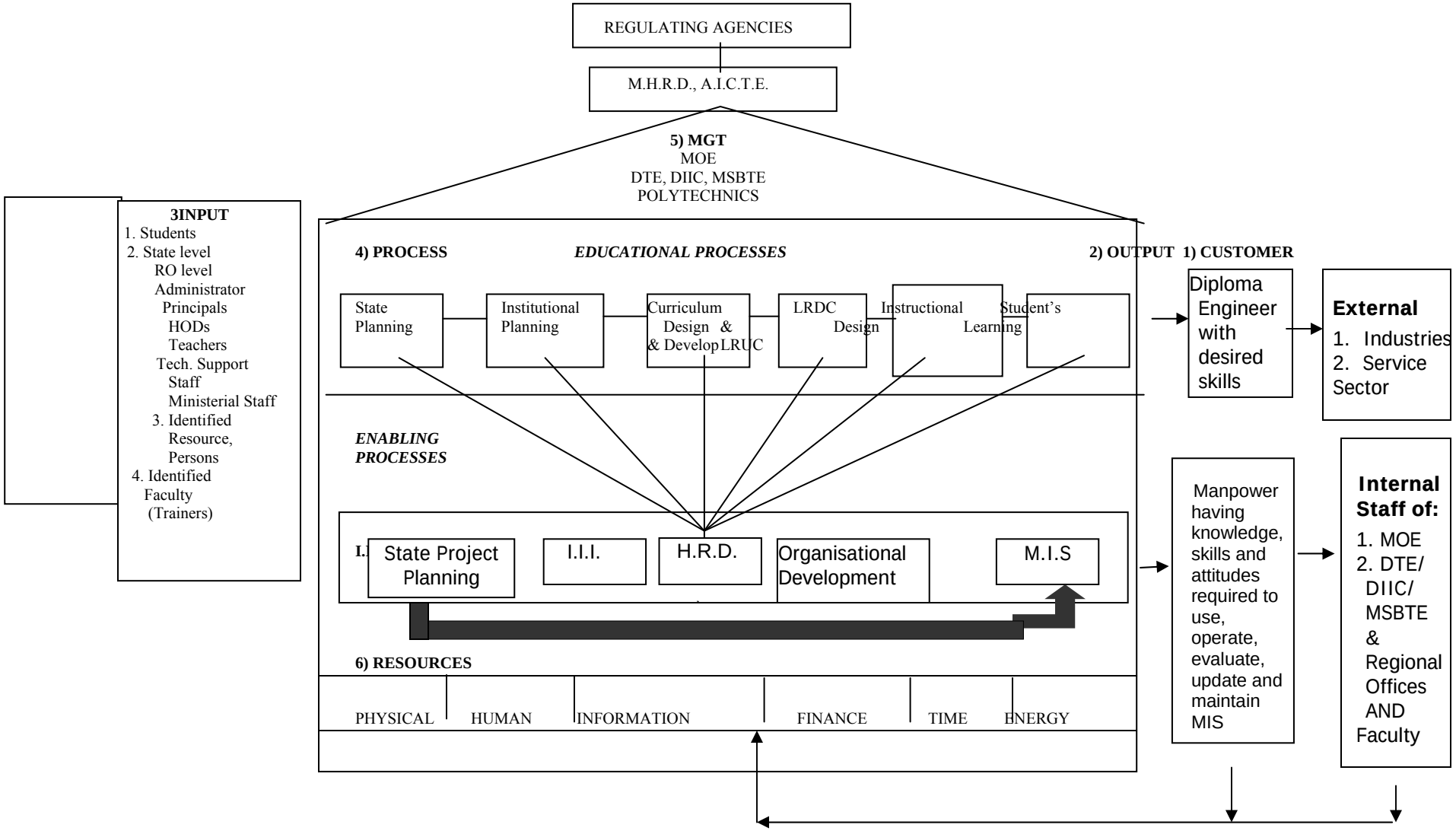


Fig 1 Systems Approach

1.4 Curriculum goals

1. To develop confidence in students by providing more exposure to industry experience and world of work at global level
2. To provide conceptual knowledge and develop analytical ability
3. To develop communication skill with good English by providing sufficient practice
4. To enhance latest technical knowledge industry interaction and media
5. To develop learning to learn skills and life skills to cope up with industrial culture
6. To impart managerial skills by providing appropriate theoretical inputs
7. To develop problem solving ability through technical projects.

1.5 DESIRED SKILLS

Industries expect from the diploma engineer the abilities and skills of general nature and specific to the job performance. The curriculum aims at developing life skills and technological skills so that the diploma pass outs would be suitable for industry. The skills are listed below:

Life Skills:

- Search information from various sources
- Develop communication ability
- Develop Presentation skill
- Work as a member of a team/group and as leader
- Collect field data
- Develop Learning to learn
- Write report for given task/work/project
- Develop computer proficiency
- Develop observation skills

Technological Skills:

Diploma engineers should possess following intellectual and motor skills in order to satisfactorily perform duties assigned to them:

A) Intellectual skills.

1. Identify the problem
2. Prepare the algorithms
3. Analyze the problem
4. Prepare the flowchart/model
5. Select hardware and software tools and technologies
6. Use of appropriate programming languages
7. Write programs
8. Test and debug computer Program
9. Diagnose the hardware faults
10. Prepare and interpret software documentation

B) Motor Skills.

1. Handle the Computer system
2. Handling trouble shooting tools
3. Assemble and disassemble computer system
4. Install hardware devices
5. Install network

1.6 Salient Changes in the curriculum:

- For First Semester Basic Science is divided into two parts- Basic Physics and Basic Chemistry. Theory examination of both parts as well as practical examination of both parts will be conducted on separate days. Sum of theory marks of both parts shall be considered for passing theory examination of Basic Science. Similarly it is applicable to practical examination. It is mandatory to appear for theory and practical examination of both parts. Candidate remaining absent in any examination of any section will not be declared successful for that exam head.
- For second semester Applied Science is divided into two sections- Applied Physics and Applied Chemistry where the theory examination of 50 marks each and practical examination of 25 Marks each will be conducted separately and the minimum passing marks for Applied Science will be the combination of both the sections. . It is mandatory

to appear for theory and practical examination of both parts. Candidate remaining absent in any examination of any section will not be declared successful for that exam head.

- The components of Development of Life Skills were taught in two semesters. In Development of Life Skills –I the topics related to personal development, such as Learning to Learn Skills, personality development, presentation skills etc. were included. In Development of Life Skills – II the topics related to Team Building, Leadership, group behavior etc. were covered. In the revised curriculum the scope of development of life skills has been broaden to include behavioral science component. Therefore the subject Development of Life Skills – II has been renamed and it is now included at V^{th} Semester in the revised curriculum under the title Behavioral Science.
- The subject of Professional Practices was introduced to integrate the skills acquired in Development of Life Skills, through technical subjects from second to sixth semester. The experience in implementing the contents of the subject shows that there are limited activities possible in second semester as the technical knowledge given to the students is very limited. Also at sixth semester the student are doing projects in which they are performing many activities included in the Professional Practices and therefore it is proposed that the subject of Professional Practices be prescribed only for three semesters vis. Third, fourth and fifth semesters.
- Introduction of Environment Studies at fourth Semester for all courses
- From the experience of implementation of Elective Subjects at V and VI semesters in last five years, it is proposed to have only one elective at the sixth semester for all courses. However the specialized courses like Medical Electronics, Electronics and Video Engineering will not have provision for electives. For elective, student will have to choose one from the given two/three subjects.
- While revising the curriculum redundant /obsolete topics/sub topics are being replaced by new/advance technology topics/sub topics.
- In Computer Engineering Group, for fourth Semester IF Computer Networks (CON) is replaced with Data Communication and Networking.
- For Fourth Semester IF, Applied Multimedia Technology Theory subject is changed to Practical.
- For Fifth semester CO, System Programming subject is included. For IF course, Information Security subject is included

- In Sixth semester, elective subjects have been included. In order to satisfy the course objectives, ONLINE examination has been introduced for the subjects Management (All branches) and Advanced Java Programming. Linux programming has been included as a practical subject for Computer engineering/ Computer Technology branch and Scripting Technology has been included as practical subject for Information Technology. Mobile Computing has been introduced for Information Technology.

2. 0 OBJECTIVES

2.1 Introduction

Objectives are the statements which describe the expected learning outcome. Such statements enable teachers to plan instructional process with appropriate resources. These objectives also provide a direction to frame proper questions to assess the learning outcome. During last decade there has been research on cognitive approach in psychology. This approach is based on biological structure of brain and meta-cognitive knowledge dimension. Important elements of this approach which form basics of learning are explained below.

2.2 Domains of Learning:

Learning is a process by which students develop relatively permanent change in mental associations through experience. This is how learning is defined by cognitive psychologists. Behavioral; psychologists define learning as a relatively permanent change in behavior.

There are following domains of learning:

A: Cognitive Domain relates to intellectual skills or abilities

B: Affective Domain relates to emotions, feelings, likes, dislikes etc.

C: Psychomotor Domain relates to manipulative skills of hands, legs. Eye-hand coordination in Engineering & Technology courses, endeavor is made to design curriculum with a focus on development of cognitive skills through classroom teaching. Whereas manipulative (psychomotor) skills are developed in workshops, laboratories & seminars where students work individually or in a group. Development of affective skills attitudes and value is supposed to be acquired through projects and co-curricular activities. These are also developed from the work culture or institutions.

How far a student has developed these abilities/skills especially from cognitive and psychomotor domains is assessed on the basis of suitable examinations. When classroom and laboratory teaching is viewed in this light, evaluation becomes an integral part of teaching – learning process.

2.3 LEVELS OF LEARNING:

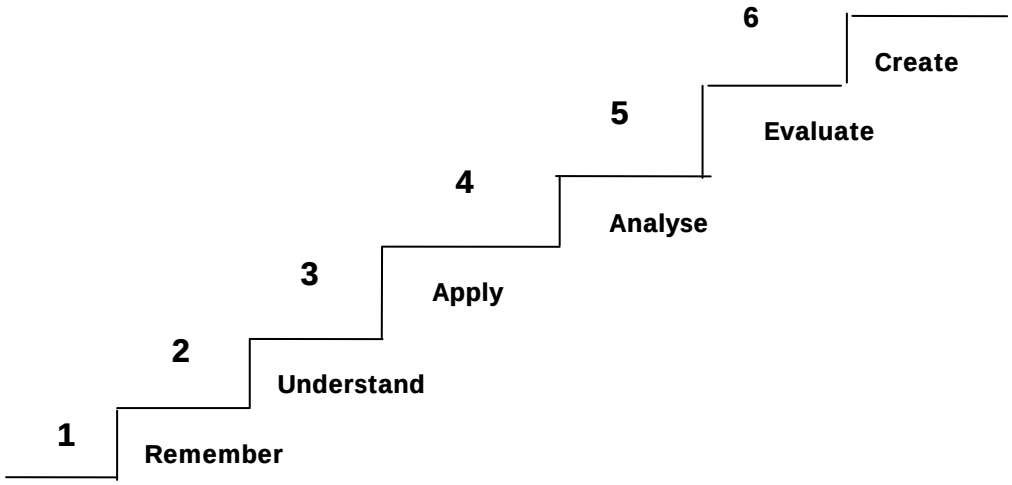
Question paper is a tool/ instrument designed to test the extent of learning of the student. Various questions set in a question paper should assess the abilities of students to respond to level of learning. Dr. Bloom a German educationist classified levels of learning in cognitive domain for the purpose of writing objectives and assessment. Dr. Bloom's revised taxonomy is based on cognitive

psychology and is two dimensional. First dimension is cognitive process dimension ad other is knowledge dimension. Details of these two dimensions are given below.

2.4.1 Cognitive Domain:

Dr. Benjamin Bloom (1956) analyzed questions asked in various examinations in American situation and proposed a hierarchical arrangement of instructional objectives (Intellectual abilities) tested by these questions.

The lowest level of cognitive learning achieved by a student is demonstrated by the recall of information that the student retrieves from his long term memory. So, the storage and retrieval of specific facts, concepts, principles, laws, definitions, properties, procedures etc. directly from memory was classified as a knowledge level objective. Thus questions testing memory of students were treated as at the lowest level of the hierarchy of intellectual abilities. The other levels of hierarchy proposed by Dr. Bloom in 1956 relate to the degree of information processing required in the brain needed to provide answer to a question. The various levels in the cognitive hierarchy proposed by Dr. Bloom in 1956 and further revised in 2001 are given below in the diagrammatic form.



Following are the details of each level which indicate the general and specific objectives. Further appropriate verbs are given which are useful in setting good questions. In this table only four levels are considered for diploma students.

Description of the Major Levels in the cognitive Domain (Bloom's Taxonomy)	Illustrative General Instructional Objectives	Illustrative verbs for stating specific learning outcomes

Remember – Knowledge is defined as the remembering of previously learned material. This may involve the recall of a wide range of material, from specific facts to complete theories, but all that is required to mind of the appropriate information. This represents the lowest level of learning outcomes in the cognitive domain	Knows common terms, specific facts, basic concepts, principles, methods & procedures	Define, describe, identify label, list, match, name, outline, reproduce, select, state
Understand – This is defined as the ability to grasp the meaning of material. This may be shown by translating material from one form to another (words or numbers) by interpreting material (explaining or summarizing), and by estimating future trends (predicting consequences or effects). Draw sketches these learning outcomes go one step beyond the simple remembering of material and represent the lowest level of understanding.	Understands fact, principles Interprets verbal material, Interprets charts, tables, graphs. Translates verbal material to mathematical formula. Estimates consequences implied in data. Justifies methods & procedures.	Convert, distinguish estimate, explain, extend, generalize, give examples; infer, paraphrase, predict, rewrite, summarize, draw labeled sketches.
Apply – Application refers to the ability to use learned material in new and concrete situations. This may include the application of such things as concepts, principles, rules, methods, laws and theories. Learning outcomes in this area require a higher level of understanding than those under the level described earlier.	Applies principles to new situations. Applies theories to practical situations. Solves mathematical problem. Construct charts, graphs Demonstrates correct usage of a procedure	Change, compile, demonstrate, discover manipulate, modify operate, predict, prepare, produce, show, solve, use.
Analyze – Analysis refers to the ability to break down material into its component parts so that its organizational structure may be understood. This may include the identification of the parts, analysis of the relationship between parts, and recognition of the organizational principles involved. Learning outcomes here represent a higher intellectual level than "understand" and apply because they require an understanding of both the content and the structural form of the material.	Recognizes unstated assumptions and logical fallacies in reasoning. Distinguishes between facts and inferences. Evaluates relevance/ adequacy of data.	Breakdown, diagram, differentiate, discriminate, distinguish, identify illustrate, infer, outline, point out, relate, select, separate, subdivide.

2.4.2 Categories of Knowledge Dimension

After considering the various designations of knowledge types, especially developments in cognitive psychology that have taken place since the original framework of Bloom's taxonomy, knowledge is categorized in 4 types – Factual , Conceptual, Procedural and Meta-cognitive.

Factual Knowledge (A) is knowledge of discrete, isolated content elements. It includes knowledge of terminology and knowledge of specific details and elements. In contrast,

Conceptual Knowledge (B) is knowledge of "more complex, organized knowledge form". It includes knowledge of classifications and categories, principles and generalizations and theories, models and structures.

Procedural Knowledge (C) is "knowledge of how to do something". It includes knowledge of skills and algorithms, techniques and methods, as well as knowledge of criteria used to determine and/or justify "when to do what" within specific fields and disciplines.

Meta-cognitive knowledge (D) is "knowledge about cognition in general as well as awareness of and knowledge about one's own cognition. It encompasses strategic knowledge, knowledge about cognitive tasks, including contextual and conditional knowledge; and self-knowledge".

Assessment is required to be done on the basis of categories of knowledge and levels of learning. Table below indicates the two dimensional grid based on Blooms Taxonomy for setting questions.

Knowledge Dimension	COGNITIVE PROCESS DIMENSION			
	1 Remember	2 Understand	3 Apply	4 Analyze
A. Factual Knowledge				
B. Conceptual Knowledge				
C. Procedural Knowledge				
D. Meta-cognitive Knowledge				

2.5 Components of Curriculum:

2.5.1 Rationale: It indicates the logical basis for the inclusion of the subject in the curriculum it also indicates the importance of the subject related to entire curriculum.

Rationale tells the students the connection of subjects related to study of higher level subjects and also the use in their job/profession.

2.5.2 Objectives: Objectives indicate what the student will be able to do/perform after he/she completes the study of the subject. It also in other words indicates the scope of the subject.

Objectives indicate what is achievable and hence give direction to the student about how to study the subject, what important things are to be observed and performed during practical.

Just as rationale indicates the use of the knowledge gained while studying the subject, objectives indicate how efficiently and effectively one can work if the objectives are fulfilled while studying the subject.

2.5.3 Learning Structure: It graphically/pictorially indicates the content of the curriculum of the subject and what is to be learnt in the subject. As you know that Cognitive Domain knowledge is divided in four components as mentioned in the Two dimensional grid. Of this Factual, Conceptual and Procedural knowledge components are identified in the curriculum of the subject along with the applications.

Facts, Concepts, Principles are used in developing procedures and applications. So these are given sequentially below procedure as Principles, Concepts and Facts in their order. Learning structure also provide an idea about how to develop the subject logically to achieve the objectives.

2.5.4 Contents: List of topics and subtopics to be included in the curriculum of the subject is given in the contents. This helps in achieving the rationale and objectives identified. Contents indicate the importance of the topics, sub topics in development of the subject and accordingly weightage in terms of Hours required to teach the subject components, so that the desired learning takes place. Marks to be allotted while testing the knowledge gained by the student are also indicated.

2.5.5 Practical's: While designing the curriculum the objectives are identified. To achieve these objectives students have to develop certain intellectual and motor skills. These skills are developed through well designed Practical. So in the curriculum the list of the skills to be developed through Practical is given. The list of Practical is so developed that after performing the Practical identified skills will be developed. Here it is necessary that the teacher gives enough opportunity to all the students to perform the practical properly to develop the skills in each one of them.

The skills will be developed if the students actually perform certain activities or tasks. Therefore it is necessary that any practical included in the curriculum necessarily involve some activities to be done by the students. So one has to think and innovate to modify the study experiments so that students will be asked to perform some activity. It could be in terms of identifying components, listing of materials used for manufacturing the components, stating importance of use of certain materials etc.

So any curriculum of a subject is so designed that it achieves the objectives of that subject as well as fulfill the objectives of the entire curriculum

3.0 CONTENT ANALYSIS

3.1 Components of Content Analysis:

As we have discussed earlier, any curriculum or syllabus of a SUBJECT given to the teacher is organized in terms of UNITS which include TOPICS or SUB-TOPICS as the case may be indicating the TIME in which it is expected to be taught to the students. Components of a topic or part thereof are analyzed here at a micro level.

Before we begin actual teaching of any topic (lesson), we must carefully and critically analyze it so that we can plan for teaching - select appropriate media, methods and techniques of teaching and arrange the suitable resources to be required. This analysis of the content of a Topic results in identification of the following components of the content:

1. Facts
2. Concepts
3. Principles (rules, laws, theories)
4. Applications
5. Procedures
6. Skills (Psychomotor Skills), and
7. Attitudes (underlying affective behaviors as quite often these are not specifically mentioned in the curriculum, still they are to be developed lesson after lesson gradually).

When we undertake the exercise of content analysis, we ourselves understand the subject fully well and at the same time we become clear as to what we are going to teach. It also gives us an idea as to which methods of teaching and media of instruction we should prepare and use and also what resources including time we will require. This analysis will also enable us to design assignments as well as how we are going to assess students learning.

Since the nature of the components of content (1 to 7) differs from one another. These are learned by the students differently as different mental processes are involved in learning these components.

The immediate implication of this varying nature of components is that these need to be taught differently and assessed differently. For example, if you look at components I to 5 all of which belong to Cognitive Domain of Learning; Component 6 belongs to Psychomotor Domain and Component 7 belongs to Affective Domain (cannot be taught as these attitudes are caught), you will find that these differ from one another. The classification of human behaviors (activities) into the above three domains of learning entails the use of entirely different methods and media of instruction. Different locations of learning (classroom, laboratories, workshops, field visits) need to be selected.

Now we will discuss these components in some detail and see how each one of these should be taught and assessed differently.

3.1.1 FACTS:

These are universally accepted and commonly understood items about which there cannot be much argument and discussion. These are required only to be informed. For example: The sun rises in east and sets in the west; names of scientists and the year in which their theories were propounded; the rules and regulations of admission and examination prescribed by the University are some of the examples of facts. Sometimes, they need not be emphasized in the class as the students already know them. But information can be passed on by word of mouth, if deemed necessary.

3.1.2 CONCEPTS:

A concept is an abstraction or an idea that permits the learner to classify a variety of related phenomena into a convenient and meaningful category. Concept of something is like a picture formation of that thing which helps in conceptualizing it. Gagne says that concept learning produces a certain fundamental change in human performance that is independent of subject or content. Concepts can be divided into the following two categories:

1. **Concrete Concepts:** those which can be seen, touched and manipulated e.g. house, book, table, chair, cat, dog, any machine or apparatus, overhead projector, chalkboard and duster.

2. **Abstract Concepts:** those which cannot be seen and touched and handled but can only be imagined e.g. force, work, fractions, decimal, bending moment, moment of inertia, friction, heat,

and induction. Teaching of concrete concepts is not that difficult because the teacher can show the object physically or its picture. On the contrary, teaching of an abstract concept offers difficulty to the teacher as well as for students to understand. These concepts can be learned by heart without understanding as children mug up Nursery Rhymes without understanding even a single word. But at the stage of higher tearing, this type of rote learning is not desirable. Adolescents (teenagers) and adults do not accept things without understanding.

3.1.3 Concept Attributes:

We identify a concept and understand it, once we are told about its qualities characteristics, and features. They are technically called concept attributes. While teaching a concept to our students we must spell out as many attributes as possible for better understanding of the concept.

Example: The Concept **of Friction**

Attributes:

1. Friction is a resistive force.
2. Frictional force acts in the direction opposite to the direction of the applied force.
3. Frictional force is more when the surfaces in contact are rough.
4. Smooth surfaces (perfect) have zero friction.
5. Frictional force is self-adjusting to a limit.

Towards the end of this Theme Paper a number of examples of concept attributes are given for your guidance.

The following questions pertaining to a concept (object or process) will be helpful in writing concept attributes:

1. What it is.
2. What are its constituent parts?
3. How it works.
4. How it is similar to and different from other known concepts.
5. What are its uses?

3.1.4 PRINCIPLES:

A principle is a statement of relationship between two or more concepts. Principles are sometimes called rules, laws or generalizations. In others words, relationship between two or more concepts which is scientific and universally true is called a Principle.

For Example: (related concepts are underlined)

1. Actions and reactions are equal and opposite.
2. Ohm's law I = V/R is a principle, where I (Current), V (Voltage), and R (Resistance) are the concepts. While teaching a principle we must recall the concepts which it involves. These concepts might have been taught in the previous lesson. As you already know, concept learning is a prerequisite to Principle learning. Thus we recall the concepts of current, voltage and resistance by asking questions to the students. Only after that we must tell the relationship among these i.e. Ohm's Law.

3.1.5 APPLICATIONS:

Whatever principles, laws and theories have been learned are only academic exercises unless these are applied to solve a practical problem. In other words, we call this application transfer of learning to a new situation. If you recall, the process of learning dealt with in Theme Paper 2, you will appreciate that the litmus test of learning having occurred is its application in a new situation or solving a new problem.

For example:

1. Ohm's law can be applied to find out the unknown quantity (voltage, current, and resistance).
2. Design of a structure can be made based on related principles and theories.
3. Principles of learning and events of instruction can be applied in ‘Designing a lesson Plan' and 'Presenting the lesson in the classroom".
4, The above principles can also be applied while preparing textbooks, workbooks, learning packages and laboratory manuals to be used by the students.

3.1.6 PROCEDURES:

While analyzing the content of a topic you might come across certain standard procedures which are prescribed to perform an operation or a given task. These procedures should be clearly identified and taught accordingly not to be left to chance. We should not pre-suppose that the students understand them. We cannot afford to take these things for granted.

For Example:

1. Procedure of setting up of an apparatus.
2. Procedure to start an engine.
3. Procedure to operate a machine (a lathe).

3.1.7 SKILLS (PSYCHOMOTOR):

A skill is an ability to perform a task expertly and well. The skilled performance; must meet a pre-specified standard of acceptable performance. A skill has the following three characteristics:

1. It represents a chain of motor responses;
2. It involves the co-ordination of hand and eye movements, and
3. It requires the organization of chains into complex response patterns.

Skills could be intellectual (thinking, understanding); interactive (communication skills) and social (socializing, mixing up with others) also. But normally when we use the word skills, it refers to psychomotor skills.

For Example:

1. Welding a butt joint,
2. Setting a theodolite at a station,
3. Making proper circuit connections, and
4. Turning a job on a lathe machine.

Laboratories and workshops of Polytechnics are the locations where these skills are developed among the students under the guidance of expert instructors *of* operators. Drill and practice are the main methods of teaching and learning these skills through model demonstrations and careful observations thereof.

Alongside developing these skills, desirable attitudes like cooperation, team work, leadership, safety, cost consciousness are also developed.

3.2 TEACHING OF CONCEPTS;

In order to teach concepts effectively the following steps have been suggested by De Cecco & Crawford (1974).

Steps Suggested:

1. Describe the performance expected of the student after he has learned the concept.
2. Reduce the number of attributes to be learned in complex concepts and make important attributes dominant.
3, Provide the student with verbal indicators (explanation).
4. Provide positive and negative examples (non-examples) of the concept.
5. Present the examples in close succession or simultaneously.
6. Provide occasions for student responses and the reinforcement of these responses, and
7. Assess the learning of the concept.

3.3 TEACHING OF PRINCIPLES:

De Cecco & Crawford (1974) has suggested the following steps for teaching principles effectively.

Steps:

1. Describe the performance expected of the student after he has learned the principle.
2. Decide and indicate which concepts or principles the students must recall in learning the new principle.
3. Assist the student in the recall of component concepts.
4. Help the student in the recall of component concepts.
5. Help the student to combine the concepts and put them in a proper order.
6. Provide for practice of the principle and for reinforcement of student responses.
7. Assess the learning of the principle.

3.4 CONCLUSION:

To sum up, it can be said that. it is essential for the teachers to develop the skills of 'Content Analysis' of their subjects. It brings content clarity amongst the teachers themselves. More importantly, Content Analysis will be a pre-requisite for writing Instructional Objectives of the topic to be taught. Teaching and learning process is bound to be effective once these crucial academic activities are undertaken.

4.0 CURRICULUM

Course Name : Computer Engineering Group

Course Code : CO/CD/CM/IF

Semester : Sixth for CO/CM/IF & Seventh for CD

Subject Title : Software Testing

Subject Code : 17624

Teaching and Examination Scheme:

Teaching Scheme			Examination Scheme					
TH	TU	PR	PAPER HRS	TH	PR	OR	TW	TOTAL
03	--	02	03	100	50#	--	25@	175

NOTE:

- **Two tests each of 25 marks to be conducted as per the schedule given by MSBTE.**
- **Total of tests marks for all theory subjects are to be converted out of 50 and to be entered in mark sheet under the head Sessional Work (SW).**

Rationale:

The complexity and size of today's software makes writing secure, bug-free code is extremely difficult, in such a situation testing of software before release is very essential. Software testing can be considered as "Quality Gate" which will pass / release only quality software.
Students will learn how to find bugs/errors in any computer program, how to plan an effective test approach, how to clearly report findings and to tell when software is ready to release. Also it introduces various levels and types of testing so that students will be able to practically apply appropriate testing method on application. It also covers manual testing as well as expanding manual test efforts with various automation tools.

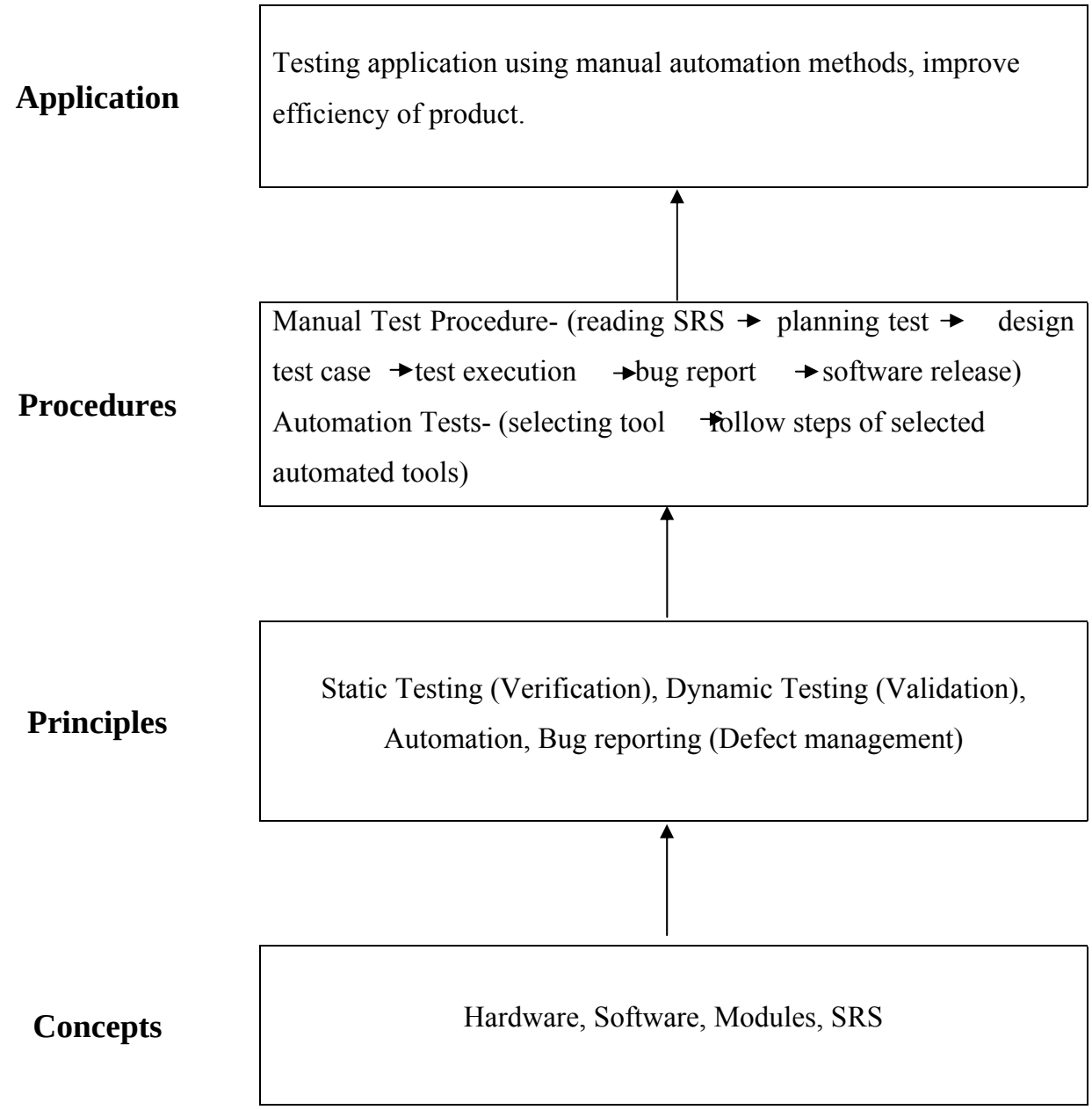
Objectives:

Students will be able to:
1. Understand how software testing fits into the software development process.
2. Learn various types and levels of software testing.
3. Develop the skills to find bugs in any type of software.
4. Learn how to effectively plan tests, communicate the bugs you find.

5. Use your new testing skill to test not just the software but also the product specification, the raw code and even the user's manual.
6. Understand STLC, test planning, test case writing and testing execution and defect management.
7. Understand the various automated testing tools to improve testing efficiency.

'G' Scheme

Learning Structure:



Topic No.	Contents	Hours	Marks
1	**Basics of Software Testing** **Objectives:** • **Understand the concept of Software Testing** • **Understand the importance of Quality Software** 1.1 Software Quality, Definition of Software Testing, Role of Testing 1.2 Failure, Error, Fault, Defect, Bug Terminology 1.3 Objectives of Testing 1.4 Test Case 1.5 When to Start and Stop Testing of Software (Entry and Exit Criteria) 1.6 Skills for Software Tester 1.7 Quality Assurance, Quality Control, Verification and Validation, V Model	04	10
2	**Types of Testing** **Objectives:** • **Understand the basic types of testing for software.** • **Differentiate White box and Black box testing** 2.1 White Box Testing : Classification of White Box Testing 1. Static Testing- Inspections, Structured Walkthroughs, Technical Review 2. Structural Testing-Code Functional Testing, Code Coverage Testing, Code Complexity Testing 2.2 Black Box Testing: Techniques for Black Box Testing Requirement Based Testing, Positive and Negative Testing, Boundary Value Analysis, Decision Tables, Equivalence Partitioning, User Documentation Testing, Graph Based Testing. Sample Examples on White and Black Box Testing	08	20
3	**Levels of Testing and Special Tests** **Objectives :** • **Understand the various levels of testing.** • **Understand some of special tests.** 3.1 Unit Testing: Driver, Stub 3.2 Integration Testing: Decomposition Based Testing - Top-Down Integration, Bottom-Up Integration, Bi-Directional Integration, Incremental Integration, Non-Incremental Integration 3.3 System Testing: Recovery Testing, Security Testing, Performance Testing, Load Testing, Stress Testing, Usability Testing, Compatibility Testing 3.4 Acceptance Testing: Acceptance criteria, Alpha Testing an Beta Testing 3.5 Special Tests: Smoke Testing and Sanity Testing, Regression Testing, Usability Testing, GUI Testing, Object Oriented	12	24

	Application Testing: Client-Server Testing, Web based Testing		

Topic No.	Contents	Hours	Marks
4	**Test Management** **Objectives:** • **Design and execute test cases.** • **Understand the Test Report Process for recommending the product Understand the process of test planning.** • **Identify resources for test plan implementation and decide the staffing for release.** 4.1 Test Planning : Preparing a Test Plan, Scope Management, Deciding Test Approach, Setting Up Criteria for Testing, Identifying Responsibilities, Staffing, Training Needs, Resource Requirements, Test Deliverables, Testing Tasks 4.2 Test Management: Choice of Standards, Test Infrastructure Management, Test People Management, Integrating with Product Release 4.3 Test Process: Base Lining a Test Plan, Test Case Specification, Update of Traceability 4.4 Test Reporting: Recommending Product Release. Matrix, Executing Test Cases, Collecting and Analyzing Metrics, Preparing Test Summary Report	12	20
5	**Defect Management** **Objectives:** • **Find, handle and report defect by using standard technique.** • **Understand the Defect life cycle.** 5.1 Introduction, Defect Classification, Defect Management Process 5.2 Defect Life Cycle, Defect Template 5.3 Estimate Expected Impact of a Defect, Techniques for Finding Defects, Reporting a Defect	06	14
6	**Testing Tools and Measurements** **Objectives:** • **Understand the shortcomings of manual testing.** • **Understand the use of automated test tools.** 6.1 Limitations of Manual Testing and Need for Automated Testing Tools 6.2 Features of Test Tool: Guideline for Static and Dynamic Testing Tool 6.3 Advantages and Disadvantages of Using Tools 6.4 Selecting a Testing Tool 6.5 When to Use Automated Test Tools, Testing Using Automated Tools 6.6 What are Metrics and Measurement: Types of Metrics, Project Metrics, Progress and Productivity Metrics	06	12
		48	**100**

List of Practical:

Sr. No.	Title of Experiments	No. of Hours
1	To Study Software Testing concepts, types and methods.	**2**
2	To study any one sample system specification and design the test cases for it. (e.g. Student information system, Library management system, Hospital management system etc.)	**2**
3	To design test cases for any application such as railway reservation form	**2**
4	To write test cases on simple calculator application.	**2**
5	To design test cases for any login form (Eg: Gmail or Yahoo login form)	**2**
6	To design test cases for mobile phone system (Eg: check battery is inserted in mobile properly, check SIM is inserted properly, check incoming and outgoing call)	**2**
7	To design test cases for notepad/WordPad/MS-Word application.	**4**
9	To design test cases for paint application.	**4**
9	To design test cases for ATM machine.	**4**
10	Using any testing tool, atomize and run test cases for notepad / WordPad.	**4**
11	Using any freeware automation testing tool, atomize and run test cases for Ms-Word application	**4**
	Total	**32**

Learning Resources:

1) Books:

Sr. No.	Author	Title	Publisher
1	Srinivasan Desikan Gopalaswamy Ramesh	Software Testing: Principles and Practices	PEARSON

2	M. G. Limaye	Software Testing: Principles, Techniques and Tools	Tata McGraw-Hill
3	Naresh Chauhan	Software Testing: Principles and Practices	Oxford

2) Testing Software:

Sr. No.	Testing Tool	Types of Tools
1	Selenium	Freeware
2	Mantis Bug Tracker	Freeware
3	IBM Rational Functional Tester	Freeware
4	MS-Excel	Commercial
5	Bugzila	----
6	Test Director	----

Note: Other possible available testing tools can be used at institute level.

3. Web Resources:

1. http://www.selenium.com
2. http://en.wikipedia.org/wiki/Test_automation
3. http://en.wikipedia.org/wiki/Software_testing#Testing_tools
4. http://www.softwaretestingsoftware.com

5. IMPLEMENTATION STRATEGY

Common guidelines for effective teaching:

- Describe learning structure of the subject to the students at the start of semester in brief.
- Prepare own subject notes.
- Refer concept structure of a topic as available in laboratory manual.
- Encourage students to co-relate topic with their academic project or any real life project.
- Ensure that structure of each model is properly learned and understood by learner.
- Give the relevance of each topic with real life applications such as E-mail, web site, e-transaction etc.

5.1 Planning of Lectures for a Semester with Content Detailing:

[The methods used to explain the contents are just guidelines. Any relevant method can be used for better understanding of students and effective teaching learning process]

Topic 1: Basics of Software Testing

Specific Objectives:(as given in curriculum)

1. Understand the concept of Software Testing
2. Understand the importance of Quality Software

Additional Specific Objectives:

- ✓ Understand Bug terminology
- ✓ Understand Importance of software Quality Assurance and control

Knowledge Category	Example/s of category	Teaching methodology
FACT	Quality, Software Quality, Failure, error, Fault, Defect Bug, , Definition of quality, Test cases	One or two case studies of software bugs from past Like(i.e. Y2K problem, or Disney Lion's king Game), Testing Definition, Need of testing to achieve Quality,
CONCEPT	Bug terminology, Software testing When to start and stop testing of software(Entry Exit Criteria)	Objectives of testing Explanation of Quality cycle, Ask students to suggest possible bugs. Entry Exit Criteria: Ref. book 1. Page 31
PRINCIPLE	Software Testing Principles , Quality Management System, Quality Assurance, Quality control, Continual Improvement, Continuous Improvement	Role of tester, explanation of(Entry Exit Criteria with diagram, Explanation of different skills of tester needed to possess Objectives of testing: Ref. book 1. Page 6 ,7 Quality assurance and Control: Ref. book 1. Page 29
PROCEDURE	Skill of testing , Verification and Validation, Role of testing, Skills of tester	Process of Verification and Validation, Skills of tester
APPLICATION	Total quality Management, Automation in software testing, V-Model	Concept of Total Quality management to achieve quality, functioning of V-Model: Ref. book 1. Page 37-42

Learning Resources:		
Books:		
Title:	**1. Software Testing – Principles and practices**	Srinivas Desikan, Gopalaswamy Ramesh, Pearson
	2. Software Testing – Principles techniques and tools	M.G. Limaye, McGraw Hill
	3. Software Testing – Principles and practices	Naresh Chauhan, Oxford
Teaching Aids: Black board, Chalk, Transparencies, Power point presentationslides(PPTs), Reference books, notes, LCD projector/Over Head Projector(OHP), Video Lectures (if made available)		
Video Lectures:	**www.H2KINFOSYS.com**	
Website:	**http://www.softwaretestingclass.com** **www.anuradhab.wordpress.com**	

Lect. No.	Topic	Topics to be covered
1	**1.1**	**Software Quality,** **Definition(s) of Quality:** • Predictable degree of uniformity, dependability at low cost and suited to market. • Degree to which a set of inherent characteristics of the product / service fulfills the requirement. • Ability to a product or service that bears upon its ability to satisfy implied or expressed need **Quality is:** • Fitness for use, i.e. Usability of software. • Conformance to specification • Judgement /of the customer /user about the attributes of a product. • Meet customer as per national/ international standards • Fulfill customers need as max as possible. **Software quality is:** the degree to which a system, component, or process meets specified requirements. **What is Software Quality?** Quality software is reasonably bug or defect free, delivered on time and within budget, meets requirements and/or expectations, and is maintainable. ISO 8402-1986 standard defines quality as "the totality of features and characteristics of a product or service that bears its ability to satisfy stated or implied needs." **Key aspects of quality for the customer include:** • Good design – looks and style • Good functionality – it does the job well • Reliable – acceptable level of breakdowns or failure • Consistency • Durable – lasts as long as it should • Good after sales service

<table>
<tr>
<td></td>
<td></td>
<td>
Value for money

<u>Goals of Software Testing:</u>
<table>
<tr>
<td rowspan="3">Software Testing:
Testing produces reliability and quality
Quality leads to customer satisfaction</td>
<td>Immediate Goals:
Bug Discovery
Bug Prevention</td>
</tr>
<tr>
<td>Long-term Goals:
Reliability
Quality
Customer satisfaction
Risk Management</td>
</tr>
<tr>
<td>Post-Implementation Goals:
Reduced maintenance cost
Improved testing process</td>
</tr>
</table>
</td>
</tr>
<tr>
<td>2</td>
<td>1.2</td>
<td>
Definition of Software Testing,
Testing is the process of executing a program with the intent of finding errors. (Myers)
• A successful test is one that uncovers an as-yet-undiscovered error. (Myers)
• Testing can show the presence of bugs but never their absence.(W.Dijkstra)
• Testing is a support function that helps developers look good by finding their mistakes before anyone else does. (James Bach)
• Execution of a work product with intent to find a defect.
<u>Role and responsibilities of Testing</u>
What are the roles and responsibilities of a Test Leader?
Involved in the planning, monitoring, and control of the testing activities and tasks.

In collaboration with the other stakeholders, devise the test objectives, organizational test policies, test strategies and test plans.
They estimate the testing to be done and negotiate with management to acquire the necessary resources.
They recognize when test automation is appropriate and, if it is, they plan the effort, select the tools, and ensure training of the team. They may consult with other groups, programmers – to help them with their testing.
They lead, guide and monitor the analysis, design, implementation and execution of the test cases, test procedures and test suites.
They ensure proper configuration management of the testware produced and traceability of the tests to the test basis.
As test execution comes near, they make sure the test environment is put into place before test execution and managed during test execution.
They schedule the tests for execution and then they monitor, measure, control and report on the test progress, the product quality status and the test results, adapting the test plan
They write summary reports on test status.

</td>
</tr>
</table>

		• Sometimes **test leaders wear different titles**, such as **test manager or test coordinator.** Whoever is playing the role, expect them to plan, monitor and control the testing work. Along with the test leaders, testers should also be included from the beginning of the projects, although most of the time the project doesn't need a full complement of testers until the test execution period. So, now we will see testers' responsibilities.
		Failure, Error, fault, Defect, Bug Terminology • **Failure:** the inability of a system or component to perform its required functions within specified performance requirements. • **Fault:** An incorrect step, process, or data definition in a computer program. • **Error:** A human action that produces an incorrect result. • An **error** can be a **grammatical error** in one or more of the code lines, or a **logical error** in carrying out one or more of the client's requirements. • Not all software errors become **software faults.** in some cases, the software error can cause improper functioning of the software. In many other cases, erroneous code lines will not affect the functionality of the software as a whole. • **A failure** is said to occur whenever the external behaviour of a system does not conform to that prescribed in the system specification. A software fault becomes a software failure only when it is "activated" **Definition:** A bug is defined in simple term as any error or mistake that leads to the failure of the product or software either due to the specification problem or due to communication problem, regarding what is developed and what had to be developed. **1. Bug Terminology: Failure, Error, Fault** The various terms related to software failure with respect to the area of application are listed as **Defect, Variance, Fault, Failure, Problem, Inconsistency, Error, Feature, Incident, Bug, and Anomaly.** i. ***Problem, error, and bug*** are probably the most generic terms used. ii. ***Anomaly, incident, and variance*** don't sound quite so negative and infer more unintended operation than an all-out failure. iii. ***Fault, failure, and defect*** tend to imply a condition that's really severe, maybe even dangerous. It doesn't sound right to call an incorrectly colored icon a fault. These words also tend to imply blame: **"It's his fault that the software failed.** iv. As all the words sound the same they are distinguished based on the severity and the area in which the software failure has occurred. v. When a program is run the error that we get during execution is termed on the basis of runtime error, compile time error, computational error, and assignment error. vi. The error can be removed by debugging, if not resolved leads to a problem and if the problem becomes large leads to software failure. vii. A bug can be defined as the initiation of error or a problem due to which fault, failure, incident or an anomaly occurs. viii. The program to find the factorial of a number given below lists few errors problem and failure in a program. **For example** 1. #include<stdio.h> 2. void main()

```
3. {
4. int i , fact, n;
5. printf("enter the number ");
6. scanf("%d",&n);
7. for(i =1 ;i <=n;i++)
8. fact = fact * i;
9. printf ("the factorial of a number is "%d", fact);
10. }
```

As in line number 4 the fact is not initialized to 1, so it takes garbage value and gives a wrong output, **this is an example of a bug**.

If fact is initialized to zero (fact = 0) than the output will be zero as anything multiplied by zero will give the output as zero. **This is a bug** which can be removed by initializing fact = 1 during initializing.

As the fact is declared as integer, for the number till 7! Will work perfectly. When the number entered is 8, the output is garbage value as the integer limit is from – 32767 to +32768, so in declaration change the initialization to long int fact;

Causes of software defects

1. Faulty requirements definition
2. Client-developer communication failures
3. Deliberate deviations from software requirements
4. Logical design errors
5. Coding errors
6. Non-compliance with documentation and coding instructions
7. Shortcomings of the testing process
8. User interface and procedure errors
9. Documentation errors

Objectives of Testing

What are software testing objectives and purpose?

Software Testing has different objectives.

The major **objectives of Software testing are as follows:**

- Finding defects which may get created by the programmer while developing the software.
- Gaining confidence in and providing information about the level of quality.
- To prevent defects.
- To make sure that the end result meets the business and user requirements.
- To ensure that it satisfies the BRS that is Business Requirement Specification and SRS that is System Requirement Specifications.
- To gain the confidence of the customers by providing them a quality product.

Test Scenario: - its design aspect.

- **Test cases** are written using **test scenario**. It describes each transaction and expected result of each transaction included in test scenario.
- Test cases are executed through test data.
- Actors (Person, system, hardware, or software) are taking part in transaction.

		(Simple, Complex, Average Actors), • Transactions represent the interactions of the actors with system under testing. • Test case characteristics: Accurate, Economical, Real, and reusable, Traceable to requirements, Appropriate, Self standing, Self-cleaning… etc.) **Test Case** **How to write a good test case?** Test case is an important document from validation of system perspective. These must be written at each stage of validation from unit test till acceptance testing. **Basic principles of writing a good test case are:** • Test case must be testable. • Tester should know what is to be done when to wait for system to do it. • Inform tester each transaction displayed/ replied by the system on screen at each step. And wait for user response. • Use simple conversational language for writing test case, which improves clarity and avoid communication losses. • Use consistent names of fields must be used in place of generic names. Any change in field name must be incorporated in test cases. • Tester should aware windows basics. • Order of the test cases must follow business scenario. Avoid time wastage, **Common Mistakes in writing test cases:** • Making test cases too long and combining two or more test cases in single test should be avoided • Incomplete, incorrect, and incoherent test cases can create confusion and frustrate testers. • Steps should be made very clear in test case steps. • Test case changes must be updated in software user interface. • Define pass/fail criteria correctly, i.e. test are successful or not, there is a defect or not? **Essential activities in Testing:** • Maintain test strategy, test plan, test scenario and test cases in a location so that they are available to concerned people when required for test artifacts. • Follow configuration management standards, and reasons for updates, Test cases used for testing must be reviewed before using them.

1.4

Attribute of test case suite are:

1. Project Name
2. Test suite ID and name
3. Version date, number, and author of test case
4. Approval and distribution date
5. Revision history with reasons for updates
6. Environment pre-requisites

Attribute for each test case are:	
a. Test pre-condition	b. Priority
c. Test sequence	d. Test data
e. Traceability to test scenario	f. Steps to reproduce
g. Test case name and number	h. Expected results
i. Types of testing	j. Actual results
k. Objectives	l. Test result
m. Valid/Invalid conditions	n. Remarks

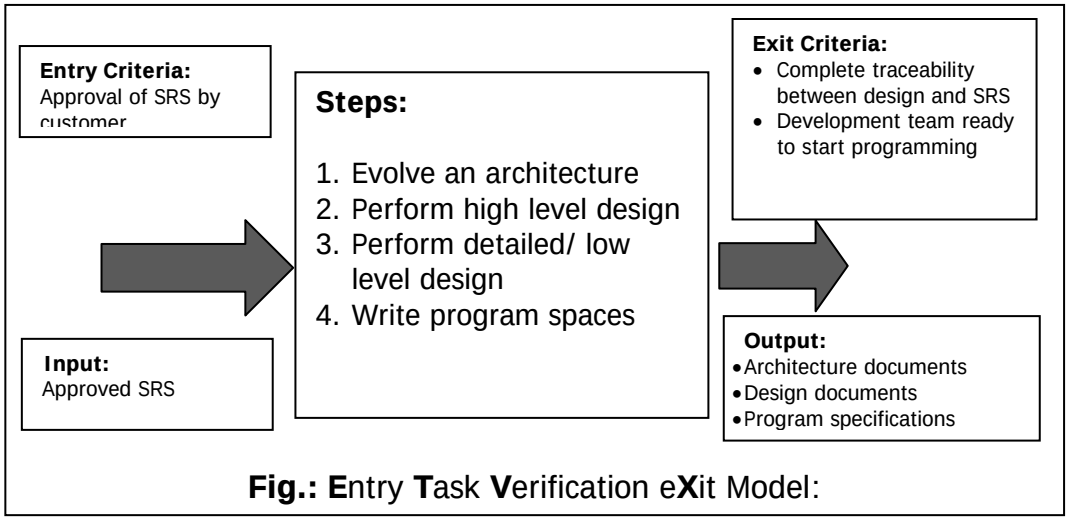


Fig.: Entry Task Verification eXit Model:

When to Start and Stop Testing of Software (Entry and Exit Criteria)

Process model is a way to represent any given phase of software development that prevent and minimize the delay between defect injection and defect detection/correction.

- **Entry criteria** –specifies when that phase can be started also included are the inputs for the phase.
- **Tasks or steps** that need to be carried out in that phase, along with measurements that characterize the tasks.
- **Verification,** which specifies methods of checking that tasks have been carried out correctly.
- **Exit criteria**, which stipulate the conditions under which one can consider the phases as done and included are the outputs for only the phase.

This is known as the Entry Task Verification exit or EVTX model which offers several advantages for effective verification and validation.

- **Clear entry criteria** make sure that a given phase does not start prematurely.

		• The verification for each phase helps to prevent defects at least minimizes.

3	1.5	**Entrance Criteria for Formal Validation Testing** • Software development is completed (a precise definition of "completed" is required. • The test plan has been reviewed, approved and is under document control. • A requirements inspection has been performed on the SRS. • Design inspections have been performed on the SDDs (Software Design Descriptions). • Code inspections have been performed on all "critical modules". • All test scripts are completed and the software validation test procedure document has been reviewed, approved, and placed under document control. • Selected test scripts have been reviewed, approved and placed under document control. **Exit Criteria for Validation Testing** • All test scripts have been executed. • All SPRs have been satisfactorily resolved. (Resolution could include bugs being fixed, deferred to a later release, determined not to be bugs, etc.) All parties must agree to the resolution. This criterion could be further defined to state that all high-priority bugs must be fixed while lower-priority bugs can be handled on a case-by-case basis. • All changes made as a result of SPRs have been tested. • All documentation associated with the software (such as SRS, SDD, test documents) has been updated to reflect changes made during validation testing. • The test report has been reviewed and approved.

Type of testing **P-Partial , F-Full**	**Unit Test**	**Component Test**	**Integration Test**	**System and Acceptance test**
Static analysis / memory leak/code complexity	**F**	**P**	--	--
Internationalization	**F**	**P**	--	--
Compatibility (Forward/ Backward)	--	**F**	--	--
Localization testing	--	--	--	**F**
Interoperability	--	--	**P**	**F**
API/Interface Testing	--	--	**F**	
Performance Testing	--	--	--	**F**
Load Testing	--	--	--	**F**
Reliability Testing	--	--	--	**F**
Functionality/ Usability	**P**	**F**	**P**	**P**
White box testing	**P**	**F**	--	--
Daily Build and smoke testing	**F**			

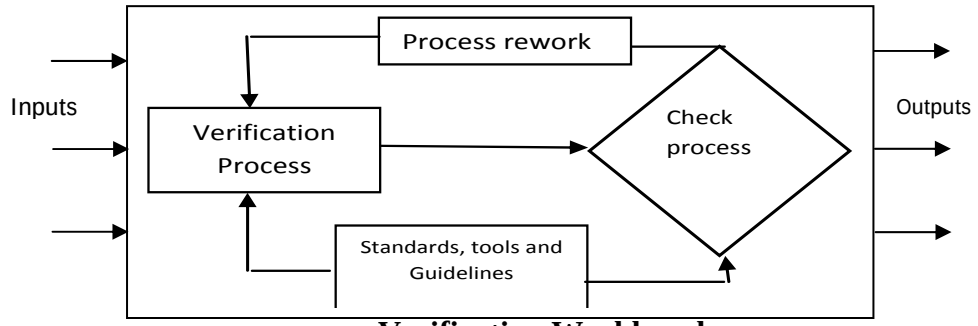
		Software quality assurance is: • The characteristics of software engineering, especially the systematic, disciplined and quantitative approach at its core, make the software engineering environment a good infrastructure for achieving SQA objectives. • The methodologies and tools that are applied by software engineering determine, to a considerable extent, the level of quality to be expected from the software process and the maintenance services. **Software quality assurance is:** • A systematic, planned set of actions necessary to provide adequate confidence that the software development process or the maintenance process of a software system product conforms to established functional technical requirements as well as with the managerial requirements of keeping the schedule and operating within the budgetary confines. 1. A planned and systematic pattern of all actions necessary to provide adequate confidence that an item or product conforms to established technical requirements. 2. A set of activities designed to evaluate the process by which the products are developed or manufactured. Contrast with: quality control.
4	**1.7**	**Verification and Validation:** **<u>Verification</u>** • Disciplined approach to evaluate whether a software product fulfills the requirements or condition imposed on them by the standard processes. • To ensure the processes and procedures defined by the customer and/or organization for development and testing are followed correctly. • It is static technique. Does not involve execution of any code, program or work product. • It helps to find root cause analysis and make corrective as well as preventive action on the defects found. **i.** It makes sure that the product is designed **<u>to deliver all functionality to the customer</u>.** **ii.** Verification is **done at the starting of the development process. It includes reviews and meetings, walkthroughs, inspection, etc. to evaluate documents, plans, code, requirements and specifications.** iii. It answers the questions like: Am I building the product right? iv. **Am I accessing the data right (in the right place; in the right way). v. It is a Low level activity** vi. Performed during development on key art facts, like walkthroughs, reviews and inspections, mentor feedback, training, checklists and standards. vii. **Demonstration of consistency, completeness, and correctness of the software at each stage and between each stage of the development life cycle.**

Advantages of verification:

- ✓ Confirms that the work product correctly as defined by an organization or customer
- ✓ It can find defects- in terms of deviations from standards, It helps to fixing defects. Avoid repetition of defects.
- ✓ It can reduce cost of finding and fixing defects it improve correction speed
- ✓ Locates defects easily.
- ✓ It can be used effectively for training people about processes and standards.

Disadvantages of Verification:

- ✓ It could not show whether the developed software is correct or not**.** Whether It shows whether the processes have been followed or not**.**
- ✓ Actual working software may not be assessed by verification as it does not cover any kind of execution of a work product, Fit for use cannot be assessed in verification.



Verification Workbench

Validation:

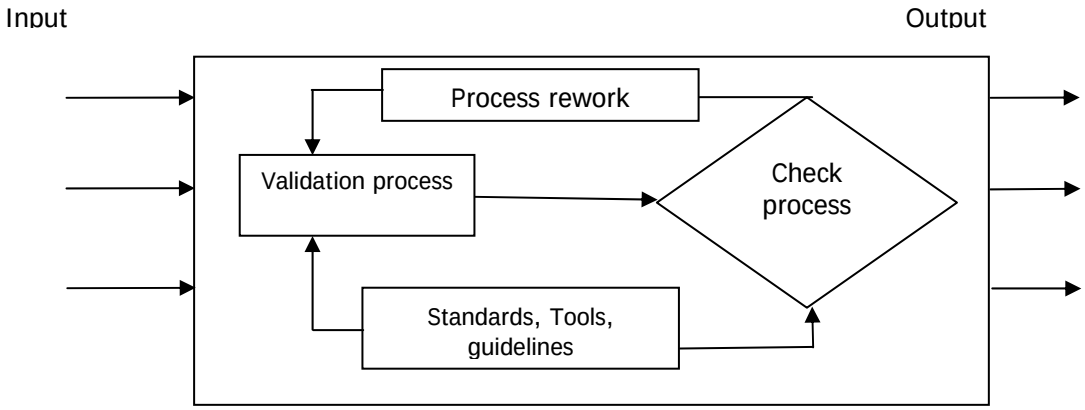
1. Determining if the system complies with the requirements and performs functions for which it is intended and meets the organization's goals and user needs.
2. Validation is done at the end of the development process and takes place after verifications are completed.
3. It answers the question like: Am I building the right product?
4. Am I accessing the right data (in terms of the data required to satisfy the requirement).
5. It is a High level activity.
6. Performed after a work product is produced against established criteria ensuring that the product integrates correctly into the environment.
7. Determination of correctness of the final software product by a development project with respect to the user needs and requirements.

Advantages of validation:

- ✓ Black box approach is used for system, integration, and acceptance testing.
- ✓ It represents actual interaction with the system without any consideration of internal structures.
- ✓ It is generally independent of the platform of development, database or any other technical aspect related to software development.

Disadvantages of Validation:

- ✓ No amount of testing can prove that software does not have defects.
- ✓ If defects are not found by conducting the defined test cases,
- ✓ There can be many more defects not captured by these test cases.

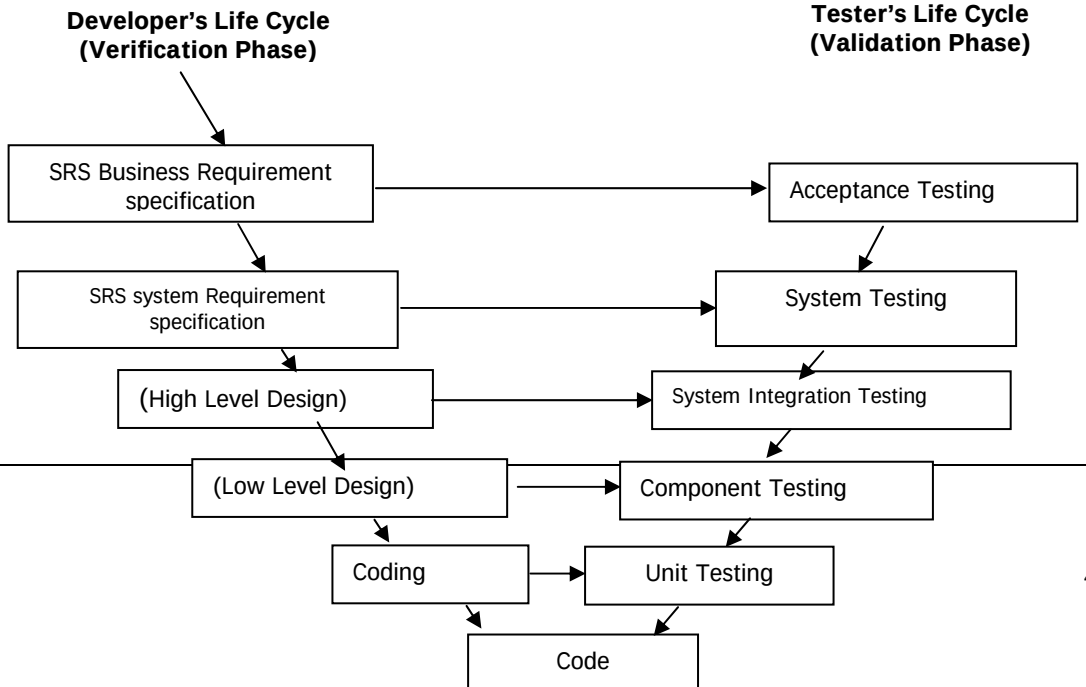


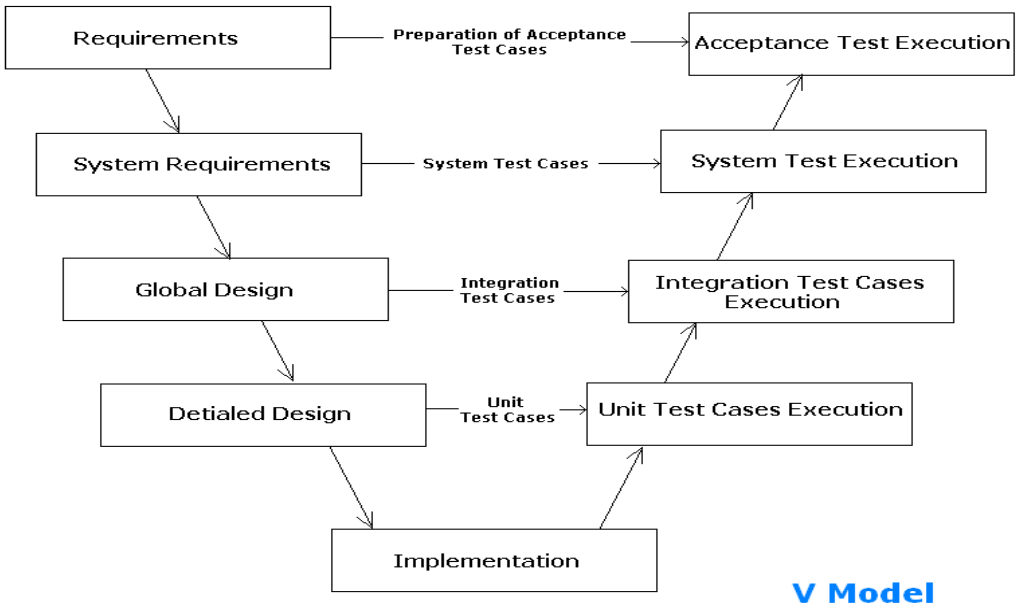
Validation Workbench

V-Model

V- Model means Verification and Validation model. Just like the waterfall model, the V-Shaped life cycle is a sequential path of execution of processes. Each phase must be completed before the next phase begins. Testing of the product is planned in parallel with a corresponding phase of development.

Diagram of V-model:





The various phases of the V-model are as follows:

Requirements like BRS and SRS begin the life cycle model just like the waterfall model. But, in this model before development is started, a system test plan is created. The test plan focuses on meeting the functionality specified in the requirements gathering.

The high-level design (HLD) phase focuses on system architecture and design. It provides overview of solution, platform, system, product and service/process. An integration test plan is created in this phase as well in order to test the pieces of the software systems ability to work together.

The low-level design (LLD) phase is where the actual software components are designed. It defines the actual logic for each and every component of the system. Class diagram with all the methods and relation between classes comes under LLD. Component tests are created in this phase as well.

The implementation phase is, again, where all coding takes place. Once coding is complete, the path of execution continues up the right side of the V where the test plans developed earlier are now put to use.

Coding: This is at the bottom of the V-Shape model. Module design is converted into code by developers.

		Advantages of V-model: • Simple and easy to use. • Testing activities like planning, test designing happens well before coding. This saves a lot of time. Hence higher chance of success over the waterfall model. • Proactive defect tracking – that is defects are found at early stage. • Avoids the downward flow of the defects. • Works well for small projects where requirements are easily understood. **Disadvantages of V-model:** • Very rigid and least flexible. • Software is developed during the implementation phase, so no early prototypes of the software are produced. • If any changes happen in midway, then the test documents along with requirement documents has to be updated. **When to use the V-model:** • The V-shaped model should be used for small to medium sized projects where requirements are clearly defined and fixed. • The V-Shaped model should be chosen when ample technical resources are available with needed technical expertise. High confidence of customer is required for choosing the V-Shaped model approach. Since, no prototypes are produced, there is a very high risk involved in meeting customer expectations.
		Revision of Topic 1

Topic 2: <u>Types of Testing</u>

Specific Objectives:(as given in curriculum)

1. Understand the basic types of testing for software.
2. Differentiate White box and Black box testing

Additional Specific Objective:

- Understand situations where White-box and Black-box testing is applicable

Knowledge Category	Example/s of category	Teaching methodology
FACT	Code/ function, Code coverage, Code , Boundary values, Decision Tables	All these facts realization with actual situation and example. Boundary situation, Concept of Decision tables (Ref. Book 1. Page 87-90
CONCEPT	White box Testing, Black Box Testing	Classification white box testing, Code review check list Ref. book 1. Page 48, Ref. book no. 1. Page 74-76
PRINCIPLE	Static Testing, Structural testing, Positive and Negative testing, User Documentation testing, Graph based testing,	**Methods of achieving static testing** • Desk checking of code • Code walkthrough • Code review and checklist • Code injection (Ref. book 1. Page 48-55) **Structural Testing** • Unit / Code testing • Code coverage testing (Statement/Path /Condition/Function coverage) (Ref. book 1. Page 56-63) **Positive and Negative Testing** (Ref. book 1. Page 82-84) **Documentation testing** (Ref. book **2**. Page 284-285) **State / Graph based testing** (Ref. book 1. Page 93-96) **User documentation Testing** (Ref. book 1. Page 99-101)
PROCEDURE	Inspections, Walkthrough, Technical Reviews, Equivalence partitioning	Importance, Activities Conducted /steps involved related these topics, Participants in conducting procedure. **Inspections, Walkthrough, Technical Reviews** (Ref. book 2. Page-167-171) (Ref. book 3. Page-190-197, 205-207) **Equivalence partitioning** (Ref. book 1. Page 90-93)
APPLICATION	Examples of white box and Black box testing	

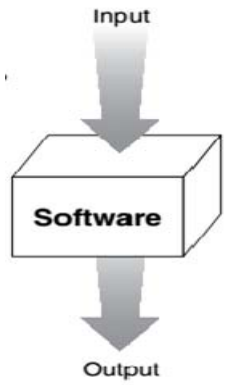
Learning Resources:

Books: Title:	**1. Software Testing – Principles and practices**	SrinivasDesikan, Gopalaswamy Ramesh, Pearson
	2. Software Testing – Principles techniques and tools	M.G. Limaye, McGraw Hill

	3. Software Testing – Principles and practices	Naresh Chauhan, Oxford

Teaching Aids:
Black board, Chalk, Transparencies, Power point presentation, slides(PPTs), Reference books, notes, LCD projector/OHP Projector

Video Lectures:	**www.guru99.com**
Website:	**http://www.softwaretestingclass.com**

Lect. No.	Topic	Topics to be covered
1	 **White Box Testing**	**2.1 White Box Testing** 1. This is also known as glass box, clear box, and open box testing. 2. In white box testing, test cases are created by looking at the code to detect any potential failure scenarios. 3. The suitable input data for testing various APIs and the special code paths that need to be tested by analyzing the source code for the application block. 4. Therefore, the test plans need to be updated before starting white box testing and only after a stable build of the code is available. 5. White box testing assumes that the tester can take a look at the code for the application block and create test cases that look for any potential failure scenarios. 6. During white box testing, analyze the code of the application block and prepare test cases for testing the functionality to ensure that the class is behaving in accordance with the specifications and testing for robustness. 7. A failure of a white box test may result in a change that requires all black box testing to be repeated and white box testing paths to be reviewed and possibly changed.

White Box testing:

Classifications of white Box testing

<table>
<tr><th>1) Static Testing</th><th colspan="4">2) Structural Testing</th></tr>
<tr><td>1.1) Desk Checking</td><td rowspan="4">2.1.1) Unit/ Code Functional Testing</td><td>2.2.1) Code coverage</td><td rowspan="5"></td><td>3.1) Code complexity</td></tr>
<tr><td rowspan="4"></td><td>2.2.2) Statement coverage</td><td rowspan="4">3.2) Cyclomatic complexity</td></tr>
<tr><td>1.2) Code Walkthrough</td><td>2.2.3) Path coverage</td></tr>
<tr><td rowspan="2"></td><td>2.2.4)Condition coverage</td></tr>
<tr><td>1.3) Code Inspection</td><td>2.2.5)Function coverage</td></tr>
</table>

2 **Static White box testing Inspections, Structured Walkthrough, Technical Review**

Static Testing:

1) This method involves principle of human reading the program to detect errors rather than computers.
2) Type of testing which requires only the source code of the project/software, not binaries or executables.
3) Does not involve executing the programs on computers but involves select people going through the code **to find out whether:**

- The code works according functional requirement.
- The code has been written in accordance with the design developed earlier in the project life cycle.
- The code for any functionality has been missed out.
- The code handles errors properly.

4) This testing can be done by human or with the help of specialized tools

Advantages of static testing:

a) Sometimes humans can find errors computers cannot.
b) Possibility of multiple perspectives by making multiple human read and evaluations of program.
c) Compares with specifications and standards, (in reactive testing reveals identifying symptoms rather than root cause) rather its **Proactive Testing**, this is done faster by human being.
d) One can find root cause for the problem.
e) Saves computer resources, as testing the code can be done before execution.

- Desk checking of code
- Code walkthrough

<table>
<tr><td></td><td colspan="2">

- Code review and checklist
- Code injection

Code review checklist:

- ✓ Data item declaration related
- ✓ Data usage related
- ✓ Control flow related
- ✓ Standard related
- ✓ Style related
- ✓ Miscellaneous
- ✓ Documentation related

</td></tr>
<tr><td>3</td><td colspan="2">Structural Testing, Code functional Testing, Code coverage Testing, Code Complexity testing:
Structural testing
Classifications:
✓ Unit/Code functional testing
✓ Code coverage
- Statement coverage
(Sequence control,
Two-way decision – If, Then, else,
Multi-way decision - switch, Loops, etc.)- while, do repeat, until and for
- Path coverage
- Condition coverage
- Function coverage
✓ Code complexity (Path independence, upper bounds), Cyclomatic complexity-Flow graph- consists of nodes and edges</td></tr>
<tr><td>4</td><td>Black box Testing ,
 

Techniques for Black box testing,</td><td>2.2 Black box testing: involves looking at specifications and does not require examining the code of a program.

➢ Done base on requirements.
➢ Addresses the stated requirements as well as implied requirements.
➢ Encompasses the end user perspectives.
➢ Handles valid and invalid inputs.

When to do it?</td></tr>
<tr><td></td><td colspan="2">Types of Black-box testing: (concepts in short)

- Requirement based testing
- Positive and negative testing
- Boundary value analysis
- Decision tables

</td></tr>
</table>

	• Equivalence partitioning • State based testing • Compatibility testing • User documentation testing • Domain testing
5, 6	**Decision Tables and Equivalence partitioning** **Decision Table:** Acts as invaluable tools for designing black box tests to examine the behavior of the product under various logical conditions of input variables.

Example decision table

	Conditions/ Courses of Action	Rules					
		1	2	3	4	5	6
Condition Stubs	Employee type	S	H	S	H	S	H
	Hours worked	<40	<40	40	40	>40	>40
Action Stubs	Pay base salary	X		X		X	
	Calculate hourly wage		X		X		X
	Calculate overtime						X
	Produce Absence Report		X				

Discuss steps in forming Decision table:

Equivalence portioning:

Parameter	Equivalence classes	Representative
Sales price	vEC1: 0 ≤ x < 15000	14500
	vEC2: 15000 ≤ x ≤ 20000	16500
	vEC3: 20000 < x < 25000	24750
	vEC4: x ≥ 25000	31800

Advantages of Equivalence partitioning are:

- ✓ This is a black box technique which divides the input domain or output domain into classes/partitions of data which test data can be derived representing the entire class.
- ✓ It reduces the probability of not finding defect belonging to a particular class.

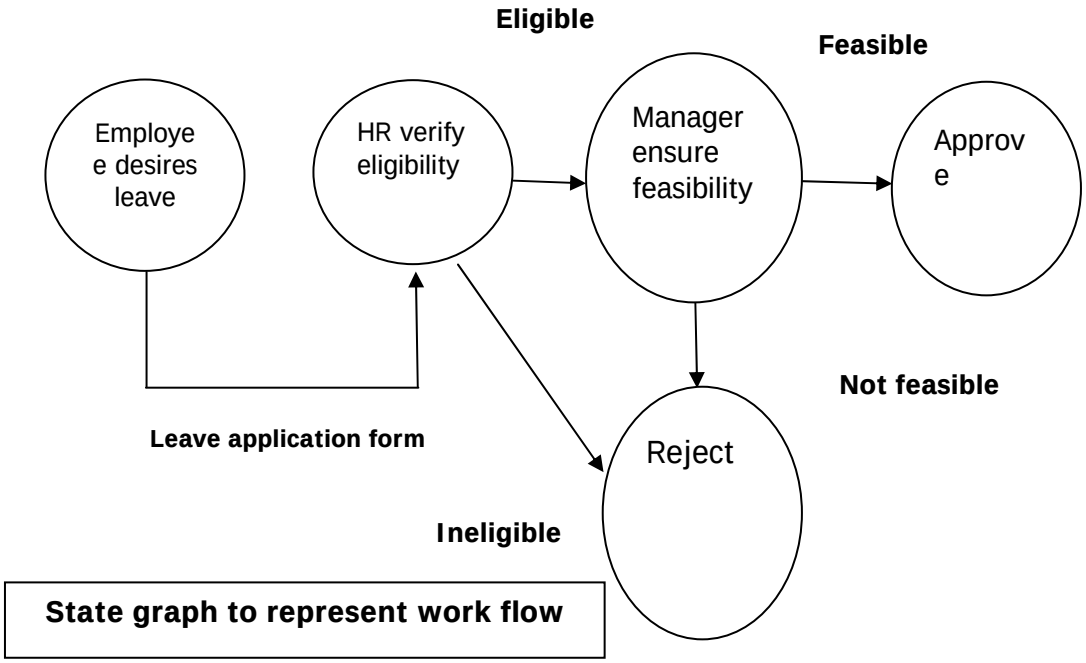
Disadvantages of Equivalence partitioning are:

- ✓ Any problem in a class may not be identified if value selected does not represent class completely.
- ✓ If class definition is not correct.
- ✓ It introduces one assumption that entire class behavior is represented by a single value, which can be wrong. When class is not a single but there are many subclasses, selection of values may not be representing all attributes of each subclass.

Examples of Equivalence partitioning are:

- ❖ Below 35 years of age(valid input)

	❖ Below 35 and 59 years of age(valid input) ❖ Above 60 years of age(valid input) ❖ Negative age(invalid input) ❖ Age as 0 (invalid input) ❖ Age as any three-digit number(valid input) **Equivalence partitions table has columns as:** • Partition definition • Type of input(valid/invalid) • Representative test data for that partition • Expected result **User Documentation testing :** • **Objectives:** **1.** To check if what is stated in the document is available in the product. **2.** To check if what is there in the product is explained correctly in the document. • It focuses on ensuring what is in the document exactly matches the product behavior , by setting in front of the system and verifying screen by screen, transaction by transaction and report by report. • It also checks for the language aspects of the document like spell check and grammar. **Benefits of user documentation testing:** ✓ Improves usability, reliability ✓ Highlighting problems over-looked during reviews ✓ High quality user documents ensures consistency of documentation and product , thus minimizes possible defects reported by customers. Reduces time for each call reduces/ minimizes support cost. ✓ When customer follows the instruction cost reduces, contributes to better customer satisfaction and better morale of support staff.

7	**Graph based Testing: Useful in following situation** • Product under test is a language processor (i.e. compiler) wherein the syntax of language automatically lends itself to state machine or a context free grammar. • Workflow modeling where, depending on the current state and appropriate combinations of input variables, specific workflows is carried out, resulting in new output and new state. • Dataflow modeling, where the system is modeled as a set of dataflow, leading from one state to another. **<u>Documentation testing checklist:</u>** • Figures and screen captures • Samples and examples • Spelling and grammar. **To design test cases for Graph based testing are:** 1. Understand the system 2. Identifying states, inputs and guards. 3. Create a state graph model of the application. 4. Verify whether state graph that we modeled is correct in all details. 5. Generate sequences of test actions.  **State graph to represent work flow**

8	**Examples of White box testing and Black box testing:** **Sample Examples on White and Black Box Testing** A simple website application as an example to explain the white and the black box testing. . The end user is simply access the website, Login & Logout; this is very simple and day 2 days life example. As end users point of view user able to access the website from GUI but inside there is lots of things going on to check the internal things are going right or not, the white box testing method is used. To explain this we have to divide this part in two steps. This is all is being done when the tester is testing the application using White box testing techniques. **Step 1) UNDERSTAND OF THE SOURCE CODE** ✓ The first & very important step is to understand the source code of the application is being test. ✓ As tester should know about the internal structure of the code which helps to test the application. ✓ Better knowledge of Source code will helps to identify & write the important test case which causes the security issues & also helps to cover the 100% test coverage. ✓ Before doing this the tester should know the programming language what is being used in the developing the application. ✓ As security of application is primary objective of application so tester should aware of the security concerns of the project which help in testing. ✓ If the tester is aware of the security issues then he can prevents the ✓ Security issues like attacks from hackers & users who are trying to inject the malicious things in the application. **Step 2) CREATE TEST CASES AND EXECUTE** ✓ In the second step involves the actual writing of test cases based on Statement/Decision/Condition/Branch coverage & actual execution of test cases to cover the 100% testing coverage of the software application. ✓ The Tester will write the test cases by diving the applications by ✓ Statement/Decision/Condition/Branch wise. Other than this tester can include the **trial and error testing**, **manual testing** and using the Software testing tools as we covered in the previous article.
	Revision of Topic 2

Topic 3: Levels of Testing and special Tests
Specific Objectives (as given in curriculum)

1. Understand the basic types of testing for software.
2. Differentiate White box and Black box testing

Additional Specific Objectives:

- Understand situations where White-box and Black-box testing is applicable

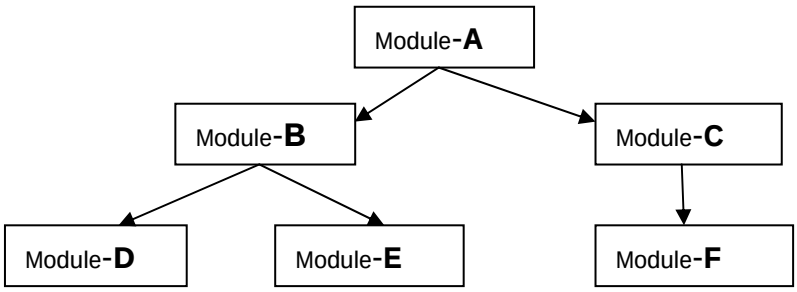
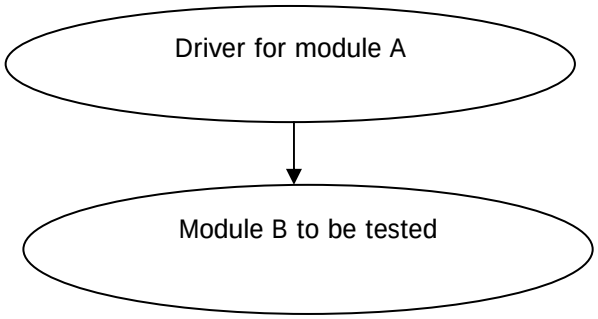
Knowledge Category	**Example/s of category**	**Teaching methodology**
FACT	Driver, stubs, Software Requirement speciation-SRS.	**Use of driver, stub in testing** **Ref. book no. 3. Page 214-217**
CONCEPT	Various levels of testing	Basic concept of different levels of testing
PRINCIPLE	Top-down, Bottom-up Integration testing,	
PROCEDURE	Unit testing, Integration testing, System testing, Acceptance testing, Security, Performance, Load Stress, Usability Test.	**Concept, Advantage(s), Disadvantages:** Unit Test, Integration Test, Ref. book no. 1. Page 108-116, Ref. book no. 2. Page 182, Ref. book no. 3. Page286-290 Incremental , Non-incremental Integration Ref. book no. 3. Page 220-226. System test and Acceptance Test Ref. book no. 1.Page 158-161, Ref. book no. 3 Page 233-234 and 244-245 Security, Performance, Volume/Load, Stress, Recovery, System, Regression, Smoke, Sanity, Compliance, Usability. Ref. book no. 1. Page no. 268-283 Compatibility Test Ref. book no. 1. Page no.260-262
APPLICATION	GUI, Object Oriented based, Web based testing, Client server testing	GUI Test, 262-268 Ref. book no. 1. Page no.260-262 Object Oriented Testing, Ref. book no. 2. Page no.298-306 Ref. book no.3. Page no.450-468 Client –server testing Ref. book no. 2. Page no. 306-316

Learning Resources:

Books: **Title:**	**1. Software Testing –Principles and practices**	SrinivasDesikan, Gopalaswamy Ramesh
	2. Software Testing –Principles techniques and tools	M.G. Limaye
	3. Software Testing – Principles and practices	Naresh Chauhan, Oxford

Teaching Aids: Black board, Chalk, Transparencies, Power point presentation slides(PPTs), Reference books, notes, LCD projector/OHP Projector	
Video Lectures:	**PPT with Sample: -**
Website:	http://www.softwaretestingclass.com

Lect. No.	Topic	Topics to be covered

1	**3.1**	**Unit Testing:** • Software product is made up of many units, each unit needed to be tested to find whether they have implemented the design correctly or not. It may involve designing and usage of stub/driver to execute units as they may not be executed alone. (Module/sub module can be treated as unit.) • It is a level of the software testing process where individual units / components of a software/ system are tested. The purpose is to validate that each unit of the software performs as designed.  **Design Hierarchy of system**  **Driver module for module-A**
2	**3.1**	**Drivers:** For Testing modules, it may require some inputs are to be received from another module, this module passes inputs to the module to be tested is not ready and under development. The module where the required inputs for the module under test are simulated for the purpose of module or unit testing is known as a Driver module. The driver module may print or interpret the results produced by the module under test. A test driver may **take inputs in the following form** and call the unit to be tested. • It may hard-code the inputs as parameters of the calling unit. • It may take inputs from the user. • It may read the inputs from file.

A test driver provides the following facilities to a unit to be tested:

- ✓ Initializes the environment desired for testing.
- ✓ Provides simulated inputs in the required format to the inputs to be tested.
- For projects if commercial drivers are not available, then specialized drivers need to be developed. It's in defence projects for special applications.

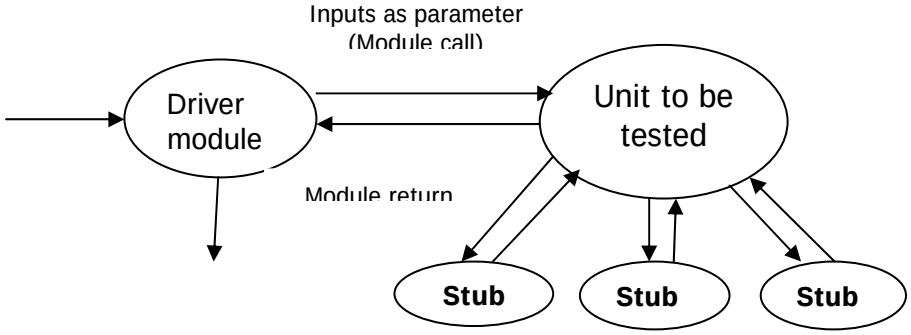
Stubs:

The module under testing may also call some other module which is not ready at the time of testing There is need of dummy modules required to simulate for testing, instead of actual modules. These are called **stubs.**

It can also be defined as **a piece of software that works similar to a unit which is referenced by the unit** being tested, its much simpler than the actual unit.

Its characteristics are :

- Its Place holder for actual module. It is a reduced implementation of the actual module.
- It does not perform any action of its own and returns to the calling unit
- It may include a display instruction as a trace message in the body of stub. The idea is that module to be called is working fine by accepting the input parameters.
- A constant must be returned from the body of stub to the calling module.
- Stub may simulate exceptions or abnormal conditions if required.



Drivers and stubs

Benefits of Stub and Drivers:

Integration Testing:

It is a level of the software testing process where individual units are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units.

It is both a type of testing and a phase of testing. It is a set of interactions all defined interactions among components needed to be tested. (How the system or modules work together)

It involves testing of many units by combining them together to form a submodule or module. It is done by developers, If its executed independently the done by testers. It ensures that the units tested independently are also going to work together correctly.

It is mainly refers to **detail** or **low-level design**.

It is testing of interfaces, i.e. Application program interfaces (API)-Internal and Software development kit(SDK) – External

Integration Methods		
Decomposition-based integration	**Call graph-based integration:** It is directed graph, nodes are modules/units	**Path-based integration:** Some path of source instruction is executed

Integration testing is necessary for the following reasons:

- It exposes inconsistency between the modules such as improper call or return sequences.
- Data can be lost across an interface.
- One module when combined with another module may not give the desired result.

documents useful in Integration testing:

- System requirement specification(SRS)
- User Manual,
- Usage scenarios containing DFDs,
- Data dictionaries and
- state charts

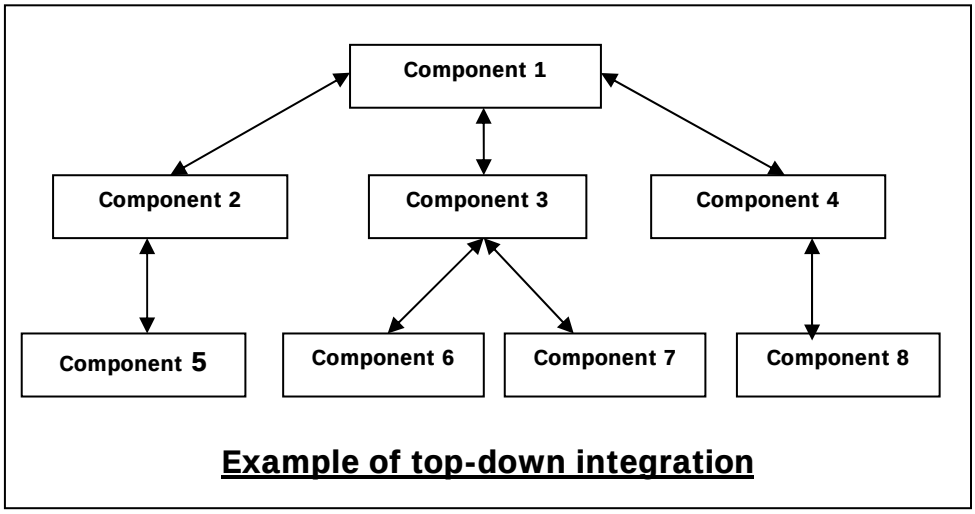
Modules for integration: criteria for selecting the modules for integration, availability, subsystems.

Strategy for integration: Well planned (generally sandwitched)

Test harness: factor useful for developing an integration test plan is the amount of dead code or

List of tasks to be tested: The functionalities and interfaces to be tested through the integration of modules must also be planned.

List of tasks NOT to be tested: The functionalities and interfaces **NOT** to be tested due to some reasons must also be planned.



Example of top-down integration

Different modules decomposition based are divided into several methodologies :

Top-down Integration:

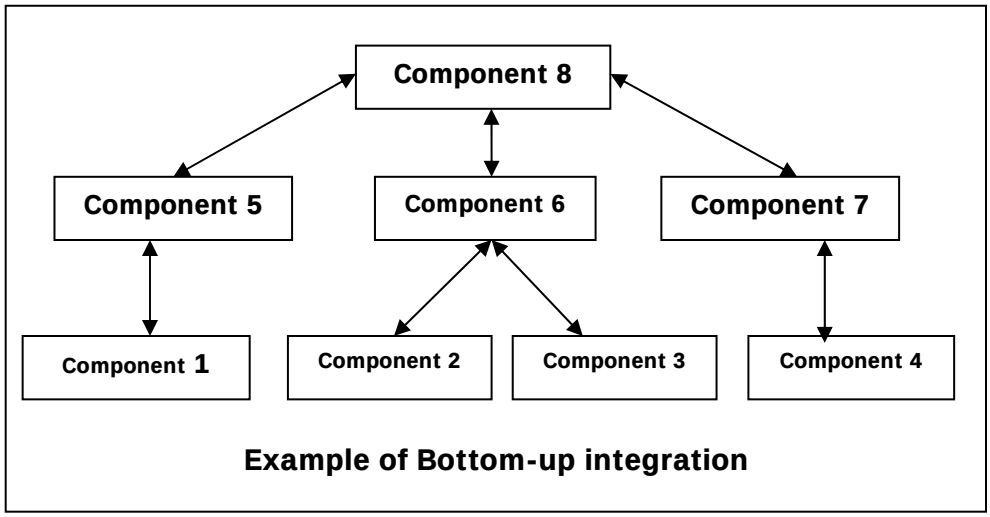
- This involves testing the topmost component interfaces with other components in the same order as from top to bottom till all components covered.
  Basically two types: Depth first search, Breadth first search.

Advantages of Top-down integration:

- Easy to determine feasibility at early stage. i.e. user interface is first.
- No. of times top-down approach does not need drivers.
- This detects major flaws in system designing by taking inputs from user
- Here design defects can be found and corrected early.

Disadvantages of Top-down integration:

- Its opposite to bottom-up approach
- Focuses on testing the bottom-up parts/units, modules and then go upwards.
- This approach tests control and data interfaces (atomic level of module).
- Use for Object oriented design and general purpose utility routines.



Example of Bottom-up integration

Bottom-up Integration

It just opposite to Top-down integration the components become available in reverse order, starting from bottom.

Advantages of Bottom-up integration:

Disadvantages of Bottom-up integration:

3	**3.2**	**Decomposition based integration.** **Bi-directional Integration:** Its mixed integration testing or sandwich testing (combination of top-down and bottom-up), Overcomes the short coming of top-down and bottom-up approaches. Here testing can start as and when modules become available. It is a vertical incremental approach. **Advantages of Sandwich integration:** **Disadvantages of sandwich integration:** **Incremental Integration:** Start with one model and unit test it, then combine the modules which have to be merged with it and perform test on both modules. **Benefits of incremental testing:** • Does not require many drivers and stubs • Interfacing errors are uncovered earlier • It is easy to localize the errors since modules are combined one by one, hence debugging becomes easier. • It is more thorough testing • Types of incremental testing are Breadth first and depth first integration. **Non-incremental Integration:** Here either all untested modules are combined together and the tested or unit tested and then combined together. Its Big-bang integration. **It cannot be adopted practically because:** • Require more work • Actual models are not interface directly until the end of the software system • It is difficult to localize the errors since exact location of bugs cannot be found easily.
4	**3.3**	**System Integration/Testing:** System integration mean that all the components of the system are integrated and tested as a single unit, Integration testing tests interfaces has two types: Components sub-system integration Final integration testing or system integration. **Advantages of system testing:** • used in Big bang models • It saves efforts and time for multi-step component integration. **Disadvantages of system testing:** • It is difficult to locate problem, to find in which interface the defect exists.

- Correcting root cause of the defect may be difficult.
- Pressure for compromising on the quality of product

Guidelines for Selection of integration method:

Sr. No.	Factors	Suggested integration method
1	Clear requirements and design	**Top-down**
2	Dynamic changing requirements, design, architecture	**Bottom-up**
3	Changing architecture, stable design	**Bi-directional**
4	Limited changes to existing architecture with less impact	**Big-bang**
5	Combination of above	**Select one of the above after careful analysis**

Recovery testing: It is the activity of testing how well the software is able to recover from crashes, hardware failure, and other similar problems.

Examples of recovery testing are:

- While application is running, suddenly restart the computer and thereafter, check the validity of application's data integrity.
- While application receives data from the network, unplug the cable, and plug-in after a while, analyze the application's ability to continue receiving data from that point, when the network connection disappeared.

Recovery testing would determine if the system can return to a well-known state and that no transactions have been compromised. **Systems with automated recovery are designed for this.**

A **checkpoint system** can also be put that records transactions and system states periodically to preserve them in case of failure. This information allows the system to return to a known state after the failure.

Security Testing:

- ✓ Safety and security issues are gaining importance due to the commercial applications on the internet and the increasing concern about privacy.
- ✓ Security is protection system that needed to assure the customers that their data will be protected.
- ✓ The efforts of security breaches could be extensive and can cause loss of information, corruption of information, misinformation, privacy violations denial of services etc.
- ✓ It is the process of attempting to devise test cases to evaluate the adequacy of protective procedures.

Types of security requirements:

- It is associated with each functional requirement.
- It has specific set related security issues to be addressed in software

implementation.

- It has some global security issues. Like in Web application

How to perform security testing?

Testers must use a risk-based approach, By identifying risks and potential loss associated with those risks in the system and creating tests driven by those risks, the testers can properly focus on areas of code in which an attack is likely to succeed. Therefore risk analysis at the design level can help to identify potential security problems and their impacts. Once identified ranked, software risks can help guide software security.

Risk management in security testing includes:

- ✓ Creating security abuse/misuse requirement
- ✓ Listing normative security requirements
- ✓ Performing architectural risk analysis
- ✓ Building risk-based security test plans
- ✓ Wielding static analysis tools
- ✓ Performing security tests

Elements of security testing:

Confidentiality, Integrity, Authentication, Authorization, Availability, non-repudiation

Performance testing:

Performance specifications are documented in performance test plan, During requirement development phase of any system development project, prior to any design effort.

It needs to know:

- ✓ Scope of performance test: system interfaces, components,
- ✓ User interfaces involved(specify peak , nominal)
- ✓ Which hardware configuration needed?
- ✓ What are time requirements for all backend batch processes, like memory use, response time, throughput and delays?
- ✓ It tests run-time performance of software as per factors.
- ✓ Important for Real-time embedded systems, as they demand critical performance requirements.
- ✓ Need to be clearly mentioned.

It is generic set of sample data is cleared in the project and because of time constraints that data is used throughout development and functional testing.

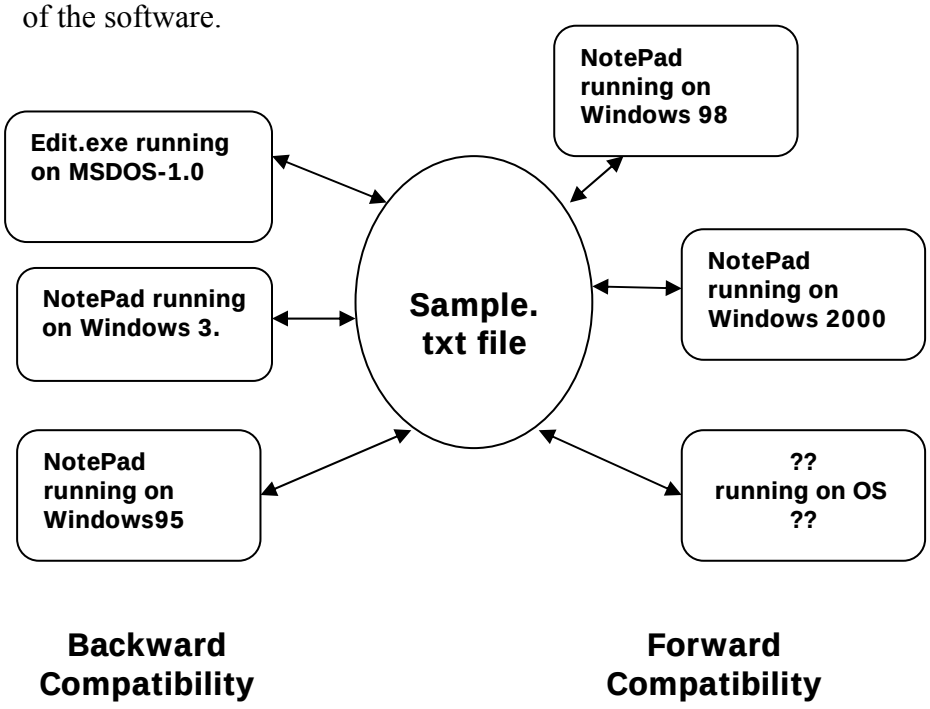
Functions performed under this test:

- Decide whether to use internal or external resources to perform tests depending on in-house expertise.
- Performance requirements from users and or business analysis.
- Develop a detailed performance test plan, test cases, workloads and environment, including all dependencies and associated timelines.

Select appropriate test tool(s), and Execute tests

6		**Load Testing:** Systems are designed and tested but they are not tested on full load (i.e. maximum values of all resources in the system to check with all its limits. It examines the behavior of a system. It is performed under controlled laboratory conditions. It is non-functional testing. **Definition:** When a system is tested with a load that causes it to allocate its resources in maximum amounts. The idea is to create an environment more demanding than the application would experience under normal workloads. Load is varied from minimum to the maximum level the system can sustain without running out of resources. Load is being increased transactions may suffer excessive delays. Load testing involves simulating real-life user load for the target application, It helps to determine how application behaves when multiple users hits it simultaneously. **<u>Examples of Load testing:</u>** • Downloading series of large files from the internet. • Running multiple applications on a computer or server simultaneously. • Assigning many jobs to a printer in a queue. • Subjecting server to a large amount of traffic. • Writing and reading data to and from a hard disk continuously. **Advantages of Load Testing:** ✓ It exposes the bugs such as undetected memory overflow and memory management bugs in your system. ✓ It increases the uptime in internet system , before it happen in an actual environment. ✓ It measures the performance of internet infrastructure. ✓ It also prevents software failure, Protect investment, increases scalability and performance. **Stress Testing:** It tries to break the system under test by overwhelming its resources in order to find the circumstances under which it will crash. It is also a type of load testing. It is designed to determine the behavior of the software under abnormal situations. In stress testing test cases are designed to execute the system in such a way that abnormal conditions. This testing is conducted to evaluate a system or component at or beyond the limits of specific requirements. **Possible areas of stress are:** • Input transactions • Disk space • Output • Communications • Interaction with users.

		It is important for Real-time systems where unpredictable events may occur, exceeds the values described in the specifications. It demands the amount of time it takes to prepare the test and amount of resources consumed during the actual execution of the test. **Examples of Stress testing:** ✓ Running several resource-intensive applications in a single computer at the same time. ✓ Flooding a server with useless email messages. ✓ Making numerous, concurrent attempts to access a single website. ✓ Attempting to infect a system with viruses, Trojans spyware or other malware. **Advantages of Stress Testing:** • Indicates the expected behavior of a system when it reaches the extreme level of its capacity. • It executes a system till it fails. • Determines the part of a system that leads to errors. • The amount of load that causes a system to fail. • Evaluates a system at or beyond its specified limits of performance. **Usability Testing:** This testing is related to a system's presentation rather than its functionality. It helps to find human factors or usability problems on the system. To adapt according to the actual work styles rather than forcing the user to adapt according to the software.(Free environment). **Usability testing identifies discrepancies between the user interfaces of a product and the human engineering requirements of its potential users.** It tests how easy software easy to us/learn and convenient to the end use. **It has following components**: Learnability, efficiency, memorability, errors, and satisfaction. **Expectation of the user from the system are:** Area of experts, group meetings, surveys, Analyses similar products, ease of use interface steps, response time help system and error messages. **Advantages of Usability testing:** • It can be modified to cover many type of testing like, Functional, system integration, unit, smoke testing. • It's very economical and effective if planned properly. • If proper resources are used, it helps to fix all problems that user may face

		even before the system is finally released to the user.it improves standard • It discovers potential bugs which are not visible to developers.
7	3.4	**Compatibility Testing** means checking whether software interacts and shares information correctly with different operating systems, hardware and software configurations available. Much software interacts with multiple CPUs, Real-time, Embedded systems which require the devices be interchangeable, removable, and re-configurable. Guidelines for Compatibility testing: Operating systems, Software/ Hardware, Conversion testing, Ranking of possible configurations, Identifying test cases and updating the compatibility test cases. There are two types of compatibility: 1. **Backward compatibility testing:** it will work with previous versions of the software. 2. **Forward compatibility testing:** it will work with Future invented versions of the software. 

Advantages of Compatibility testing:

- Truly dynamic test reveals workability and stability of software
- Efficient test ensures real compatibility among different computing environments.
- Helps to enhance image and reputation of software development firm to supply world-class software.

Disadvantages of Compatibility testing:

- ✓ Lead to rise in cost of production and time.
- ✓ Tests may be delayed which prolongs delivery of product.
- ✓ Conducting these tests will be time-consuming as need to perform various kinds of hardware and software.

Acceptance Testing:

This is a phase after system testing that is normally performed by customer or representative of the customer. They defines set of test cases that will be executed to quality and accept the product. It tests the acceptance criteria defined during the requirement phase of project. This test provides feedback to the development team.

It is formal testing conducted to determine whether a software system satisfies its acceptance criteria

This test is designed to:

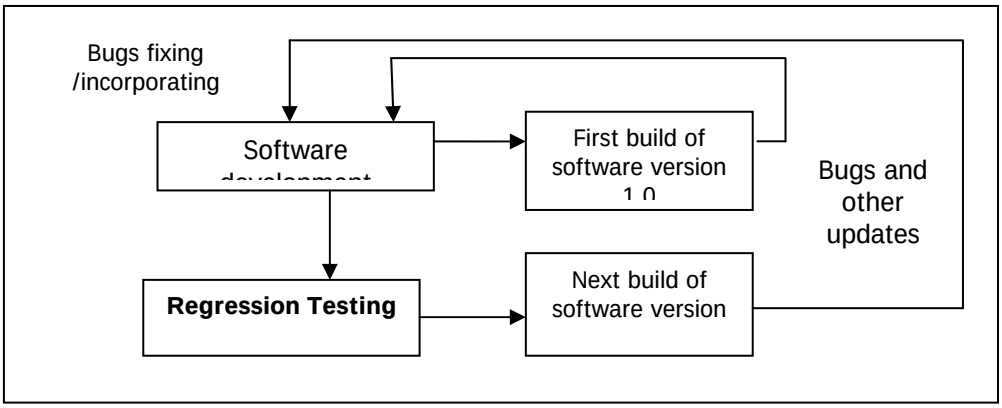
- Determine whether the software is fit for the user.
- Making users confident about product.
- Determine whether a software system satisfies its acceptance criteria.
- Enables the buyer to determine whether to accept the system or not.

Development team and the customers should work together to make sure:

- ✓ Identify interim and final products for acceptance, criteria and schedule.
- ✓ Plan how and by whom each acceptance activities will be performed.
- ✓ Schedule adequate time for the customer to examine and review the product.
- ✓ Prepare acceptance test plan.
- ✓ Perform formal acceptance test at delivery.
- ✓ Make decision based on the results of acceptance testing.
- ✓ Types of Acceptance Testing are ALPHA and BETA TESTING.

8		**Alpha testing:** It is the test period during which the product is complete and usable in a test environment, but not necessarily bug-free. It is a finale to get verification from the customer that the final development stage is coherent. **It has controlled environment.** **Features of Alpha testing:** 1. Outside users are not directly involved. 2. White box and Black box testing is used. 3. Performed at Developers site. **Alpha testing is performed for following reasons:** • To give confidence to that the software is in a suitable state to be seen by the customers. • To find bugs that may only be found under operational conditions any major defects or performance issues should be discovered in this stage. **Entry criteria to Alpha Test:** • All bugs on primary platforms are fixed/ verified. • High bugs on primary platform are fixed / verified. • 50% medium bugs on primary platforms are fixed / verified. • All features have been tested on primary platforms. • Performance has been measured / compared with previous releases. • Usability testing and feedback • Alpha sites are ready for installation. **Exit Criteria from Alpha Test:** • After alpha test : • Get responses and feedbacks from customers. • Prepare a report of any serious bugs being noticed. • Notify bug issues to developers. **Beta Testing:** After alpha testing is complete, development enters the beta phase. In this phase product should be complete and usable in a product environment. Versions of the software are known as beta-version, these are released to limited audience outside the company.(Customer's site). **It has uncontrolled environment.** **Guidelines for beta testing:** • Don't expect to release new builds to beta testers more than once every two weeks. • Don't plan beta test with fewer than four releases. • Even a small change is feature during beta process, process goes to the beginning of eight weeks and you need another 3-4 releases. **Entry Criteria to Beta Test:** • Positive responses from alpha sites. • Customer bugs in alpha testing have been addressed. • There are no fatal errors which effect on functionality of software.

		• Secondary platform compatibility testing is complete. • Regression testing corresponds to bugs fixes has been done. • Beta sites are ready for installation. •
9	**3.5**	**Smoke testing:** Smoke tests are required to ensure that the application is stable enough for testing, ensuring that the system has all the functionality and is working under normal conditions. This testing does not identify faults but establishes confidence over stability of system. Operations like Logging in addition, deletion of detection of records smoke test need to be a sub set of regression testing suite. It is time consuming and human intensive to run. It ensures major functionality is working proper. **Smoke testing consists of :** • Identifying the basic functionality that a product must satisfy. • Design test cases to ensure that these basic functionality work and packaging them into suite. • If this suite fails, changes or rollback of changes to the state where the smoke test suite succeeds. **Sanity Testing:** • This testing is performed to test the major functionality or behavior of the software application. • This testing is done after through regression testing is over. It ensures that the defects fixes or changes after regression testing does not break the core functionality of the product. • It follows deep and narrow approach with detailed testing of some limited features (particular component). • It is dene with intent to verify that end user requirements are met or not. • It is non-scripted. • It finds uncovered testing issues. • It increases level of confidence of testers. It is set of acceptance test.
10		**Regression testing:** (Video lecture will be available on www.unicomsoftlabs.com, http://interpsingh.blogspot.com) It is selective testing of a system or component to verify that modifications have not caused unintended effect and the system or component still complies with its specified requirements. It is used to ensure that bug-fixes and new functionalities introduced in new version of software do not adversely affect the correct functionality from previous version. It is the software maintenance task performed on a modified program to instill confidence that changes are correct and have not adversely affected the unchanged portion of the program. **Objectives of Regression Testing:** ✓ It finds other related bugs

		✓ It tests to check the effect on other parts of the program. **Regression testing produces Quality software.** • Validate the parts of software where changes occur. • It validates parts of software which may be affected by some changes but otherwise unrelated. • It ensures proper functioning of the software, as it was before changes occurred. • It enhances quality of software, as it reduces the high risk bugs.  **Software maintenance:** Corrective maintenance, Proactive maintenance, Perfective maintenance, preventative maintenance, rapid iterative development, First step on integration
11		Graphics User Interface Testing (GUI) Testing is important phase of testing important part of application along functionality as effects on usability. It includes following: ✓ All colors used for background, control colors, and font color have a major impact on users. ✓ User must able to identify entities on the screen correctly and efficiently. Like wrong color combinations and bright colors may increase fatigue of users. ✓ All words, Fonts, Alignments, scrolling pages up and down, navigations for different hyperlinks and pages, scrolling reduce usability. ✓ Error messages and information given to users must be usable to the user. ✓ Reports and outputs produced, readability issue, paper size on printer. Screen layouts, types of controls on single page are important. ✓ No. of images on page or moving parts on screen may affect performance. These are high-priority defects. It has direct relationships with usability testing, look, and feels of an application, It affects emotions of users and can improve acceptability of an application. **Advantages of GUI Testing:** ✓ Good GUI improves feel and look of the application; it psychologically

accepts the application by the user.

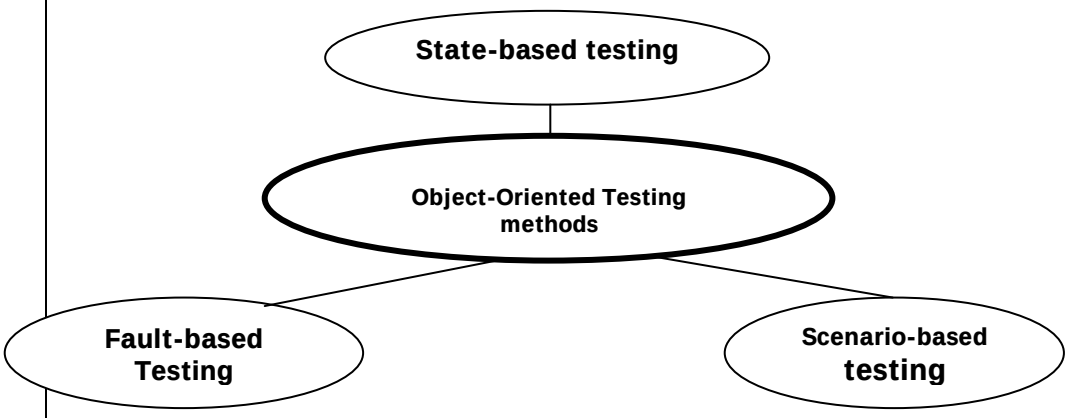
- ✓ GUI represents a presentation layer of an application due to better experience of the users.
- ✓ Consistency of the screen layouts and designs improves usability of an application.

Disadvantages of GUI Testing:

- When no. of pages is large and number of controls in a single page is huge.
- Special application testing like those made for blind people or kids below age of five may need special training from testers.

Object Oriented Testing:

- Made major change in development methodologies.
- Objects at several instances giving benefits of optimization, reusability and flexibility.
- Improves productivity with good maintainability of an application.
- many effective approaches to test object oriented software.
- Here the cost of finding and correcting bugs is always higher.
- Testing activities can begin and proceed in parallel with concept definition , object oriented architecture and designs
- Currently switching to object oriented paradigm the importance of object oriented software is increasing.



These three methods of Object oriented application testing :

1. State based Testing: used where the methods of class are interacting properly with each other. Testing seeks to exercise, State transaction diagrams are useful for performing this testing
2. Fault-based testing: used to determine a set of plausible faults. It starts by examining the analysis and design models of object oriented software as these models may provide an idea of problems in the implementation of software.
3. Scenario-based testing: used to detect errors that are caused due to incorrect specifications and improper interactions among various segments of the software, which lead to incorrect outputs that can cause

		malfunctioning of some segments of the software.
12		**Client server Testing:** The client server is improvement in standalone application, where there are several clients communicating with the server. There are many advantages of client server over stand-alone applications. It can be assessed by no. of users to work with software at a time. **There are two (three) parts**, client – server system, client, server (and network). • Multiple users can access the system at a time and they can communicate with the server. • Configuration of client is known to the server with certainty. • Client and server are connected by real connection. **Testing approach of client-server system:** • **Component Testing:** one need to define the approach and test plan for testing client and server individually. When server is tested there is need of a client simulator, whereas testing client a server simulator, and to test network both simulators are used at a time. • **Integration testing:** After successful testing of server, client and network, they are brought together to form network testing. • **Performance testing:** System performance is tested when no. of clients are communicating with server at a time. Volume testing and stress testing may be used for testing, to test under maximum load as well as normal load expected. Various interactions may not be used for stress testing. • **Concurrent Testing:** It is very important testing for client-server architecture. It may be possible that multiple users may be accessing same record at a time, and concurrency testing is required to understand the behavior of a system. • **Disaster Recovery/ Business continuity testing:** When the client server application are communicating with each other , there exit a possibility of breaking of the communication due to various reasons or failure of either client or server or link connecting them. The requirement specifications must describe the possible expectations in case of any failure. • **Testing for extended periods:** In case of client server applications generally server is never shutdown unless there is some agreed Service Level Agreement (SLA) where server may be shut down for maintenance. It may be expected that server is running 24X7 for extended period. • **Compatibility Testing:** Client server may be put in different environments when the users are using them in production. Servers may be in different hardware, software, or operating system environment than the recommended. Other testing such as security testing and compliance testing may be involved if needed, as per testing and type of system. **Web based testing:** Web application is further improvement in client-server applications where the clients can communicate with servers through virtual connectivity. Multiple server networks can be accessed at a time from the same client. It improves communication between people at different places significantly. • **Component Testing:** Here client, server as well as network testing is also required

		• **Integration testing:** In addition to this there are several special testing involved in web application. Following are some types. • **Performance testing:** System performance is tested when no. of clients are communicating with server simultaneously. • **Concurrency Testing:** It may be possible that multiple users may be accessing same records at a time and requires understanding behavior of the system as no. of users is very large. • **Disaster Recovery/ Business continuity testing:** Requirement specifications must describe the possible expectations of the system behavior in each case of any failure. MTTR (Mean Time To Repair).and MTTB (Mean Time Between Failures) are very important tests for web applications. • **Testing for extended periods:** • **Compatibility Testing:** **Mobile application testing (PDAS-Pocket device Applications**): small in size to test. Battery capacity is limited, • Bandwidth available with Bluetooth and Wi-Fi may be another challenge faced by PDAS as it is much less that bandwidth availability by other means like optical cables. • **Interface design of PDAS:** • **eBUSINESS/ eCOMMERCE Testing: Also incorporating legal standards.** **Agile development Testing and Data warehousing testing:**
		Revision of chapter 3

Topic 4: Test Management.

Specific Objectives (as given in curriculum)

1. Design and execute test cases.
2. Understand the Test Report Process for recommending the product
3. Understand the process of test planning.
4. Identify resources for test plan implementation and decide the staffing for release.

Additional Special Objectives:

- Understand the Test Management Process and Test Planning process.
- Design the Test Summary report after execution of the Test Cases.

Knowledge Category	Example(s) of Category	Teaching Methodology
FACT	Scope Management, Matrix	Deciding Feature to be Tested/ Not Tested; (Refer book no.1. Page 352) Use of appropriate PPT; Use of Video- **https://www.youtube.com/watch?v=bp3MIYUk16w**
CONCEPT	Test Management	Test Infrastructure Management, Test People Management, Integrating with Product Release**;** (Ref. book no. 1. Page 369-373) Use of appropriate PPT **Refer following Link for Test Management-** **http://en.wikipedia.org/wiki/Test_management** **http://www.softwaretestinghelp.com/test-data-management-techniques/**
PROCEDURE	Test Process	Base Lining a Test Plan, Test Case Specification, Update of Traceability; (Ref. book no. 1 Page 374 & 375) Use of appropriate PPT; **Refer following Link for Test Process –** • **http://www.softwaretestingstandard.org/part2.php** • **http://www.slideshare.net/tokarthik/Test-Process**
APPLICATION	Test Planning, Test Reporting	Preparing a Test Plan, Setting Up Criteria for Testing, Deciding Test Approach, Identifying Responsibilities, Executing Test Cases, Preparing Test Summary Report. (Ref. book no. 1. Page 374 & 375) **Refer following Link for Test Plan format-** • **http://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/** • **http://blog.bughuntress.com/automated-testing/7-tips-on-how-to-write-a-test-plan**

Learning Resources:

Books:

Title:	**1. Software Testing – Principles and practices**	Srinivas Desikan, Gopalaswamy Ramesh, Pearson
	2. Software Testing – Principles techniques and tools	M.G. Limaye, McGraw Hill
	3. Software Testing – Principles and practices	Naresh Chauhan, Oxford

Teaching Aids:
Black board, Chalk, Transparencies, Power point presentationslides(PPTs), Reference books, notes, LCD projector/OHPProjector

Video Lectures:	**PPT with Sample: -** 1. **http://www.slideshare.net/oanafeidi/test-management-introduction** 2. **http://www.slideshare.net/WilliamEchlin/principles-of-test-management-16546230?related=2** 3. **https://www.youtube.com/watch?v=f3U6V_5HEhk**
Website:	1. **http://www.testmanagement.com/** 2. **http://en.wikipedia.org/wiki/Test_Management_Approach** 3. **http://www.testingeducation.org/course_notes/amland_stale/cm_200212_exploratorytesting/exploratorytesting_2_test_management_and_techniques.pdf** 4. **http://sa.inceptum.eu/sites/sa.inceptum.eu/files/Content/Test%20management%20best%20practices.pdf** 5. **http://www.etestinghub.com/**

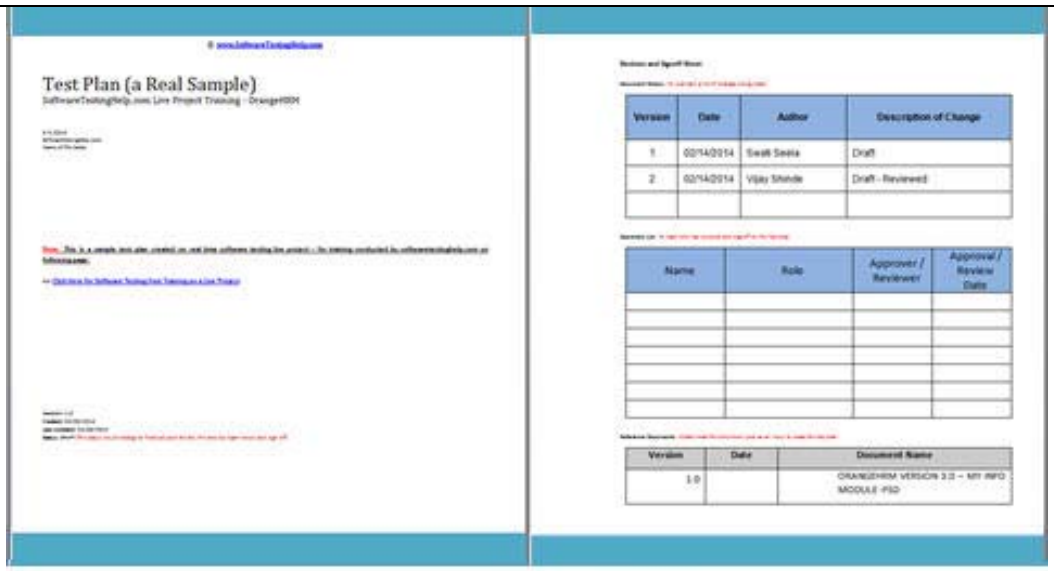
Lect. No.	**Topics**	**Topics to be covered**
1	**4.1**	**Test Planning** **Preparing a Test Plan** (Ref. book no. 1. Page 352) While Preparing a Test plan, it should acts as execution, tracking and reporting of the entire testing project and it should cover • The Scope of Testing, • How the testing is going to be performed, • What Resources are needed for testing, • The TimeLine,

- Risk factor

No.	Test Plan	Descriptions
01	Test Plan ID:	Unique No. or Name e.g. PTP_00001
02	Introduction:	Introduction about Project description
03	Test Items:	Modules /Sub modules/ Functions /Features / etc.
04	Features to be tested:	What features are going to be tested in the modules/functions etc.
05	Features not to be tested:	What features are not to be tested?
No. (03), (04) & (05) decide which modules to be tested [What to test?]		
06	Feature pass or fail criteria:	Pass or fail description [Environment is good] [After testing conclusion]
07	Suspension criteria:	Specify the criteria to suspend while doing testing activity
08	Test Environment:	Software & Hardware to be tested for above features
09	Test Deliverables:	It's a document prepared by tester to give the overall deliverable status)
10	Testing Task:	Required tasks to do before start every feature testing
Above (06) to (10) specifies How to test?		
11	Staff & Training:	Selected test Engineers & training requirements for them
12	Responsibilities:	Responsibilities of each team member.
13	Schedule:	Date & Time of the testing modules
14	Risk & Mitigation:	Possible risks & solution to overcome it.
15	Approvals:	Signatures from the Test plan authors & Project Manager / Quality Analyst

PPT refer from:
https://cs.uwaterloo.ca/~palencar/cs447/lectures/Lecture20.ppt

Video refer from:
http://www.powershow.com/view/86ae9-ODExY/Test_Plans_powerpoint_ppt_presentation



© www.SoftwareTestingHelp.com

- **Scope Management**

(Ref. book no.1. Page 352 to 354, Chapter 15 Page)
Scope Management pertains to specifying the scope of a project. For testing, Scope Management entails,

1. Understanding what constitutes a release of a product;
2. Breaking down the release into features;
3. Prioritizing the features for testing;
4. Deciding which feature will be tested and which will not be;
5. Gathering details to prepare foe estimation of resources for testing.

The following factors drive the choice and prioritization of features to be tested.

- Feature that are new and critical for the release
- Feature whose failures can be catastrophic
- Feature that are expected to be complex to test
- Features which are extensions of earlier features that have been defect prone

Refer following Link for Scope Management-

- **http://en.wikipedia.org/wiki/Scope_(project_management)**
- **http://glossary.tenrox.com/Scope-Management.htm**
- **http://www.tutorialspoint.com/pmp-exams/project_scope_management.htm**

Video refer from:

		• **http://www.powershow.com/view/168a7d-NmFlY/Chapter_4_Project_Scope_Management_powerpoint_ppt_presentation** • **http://www.powershow.com/view/26035-ZmI5O/Chapter_5_Project_Scope_Management_powerpoint_ppt_presentation** **Deciding Test Approach** (Ref. book no.1 Chapter 15 Page 354) Deciding Test Approach include identifying 1. What type of testing would you use for testing the functionality? 2. What are the configurations or scenarios for testing the features? 3. What integration testing would you do to ensure these features work together? 4. What localization validations would be needed? 5. What "non-functional" tests would you need to do? **Refer following Link for Deciding Test Approach-** • **http://www.tutorialspoint.com/software_testing_dictionary/test_approach.htm** • **http://en.wikipedia.org/wiki/Test_strategy** **Video refer from:** • **https://www.youtube.com/watch?v=JiOQSvI6Bqo** • **https://www.youtube.com/watch?v=62jVoWzFcRA**
2		**Setting Up Criteria for Testing** (Ref. book no.1**,** Chapter 15 Page 355) A test cycle or a test activity will not be an isolated, it is a continuous activity that can be carried out at one go. Some of the typical criteria include, 1. Encountering more than a certain number of defect 2. Hitting a show stoppers that prevent further progress of testing 3. Developers releasing a new version which they advise should be used in lieu of the product under test. **Identifying Responsibilities, Staffing and Training Needs** (Ref. book no.1, Chapter 15 Page 355) Identifying Responsibilities, Staffing and Training Needs addresses the aspects of test planning i.e. who part of it. A testing project require different people to play different roles, So the different role definitions should 1. Ensure clear accountability for a given task; 2. Clearly list the responsibilities for various functions to various people; 3. Complement each other; 4. Supplement each other; **Refer following Link for Identifying Responsibilities, Staffing and Training Need -** • **http://www.staffingpractices.soe.vt.edu/staffdev.htm**

Responsibilities

17. Responsibilities

Identify groups responsible for managing, designing, preparing, executing, witnessing, checking and resolving that will help the whole team to deliver a quality full website.

Role	Responsibility	Candidate	Timing
Project Manager	Managing the total implementation	Mr. A	Start to end of the project
Test Manager	➢ Create Test plan ➢ Identify test data ➢ Execute test conditions and mark-off results ➢ Prepare software error reports ➢ Administrate error measurement system	Mr. X	All, part-time
Test Planner	➢ Ensure Phase 1 is delivered to schedule and quality ➢ Produce expected results ➢ Report progress at regular status reporting meetings ➢ Manage individual test cycles and resolve tester problems.	…… …… ……	All, part-time
Test Engineer	➢ Designing the tests ➢ Creating the test procedures ➢ Creating the test data ➢ Executing tests ➢ Preparing Bug reports	…… …… …… ……	Testing time
SQA Project Leader	➢ Ensure delivered schedule and quality of the project ➢ Regularly review testing progress ➢ Manage risks issues relating to System Test Team ➢ Provide resources necessary for completing system test	……	All, part-time
Developer	➢ Ensure Unit Testing ➢ Review each module before merging	…… …… ……	Development time

3 **Resource Requirements**

(Ref. book no.1 Chapter 15 Page 356)

Project Manager should provide estimates for the various hardware and software resource required. Some of the following factors need to be considered.

1. Machine configuration,
2. Overheads required by the test automation tool,
3. Supporting tools such as Compiler, Test data generator, Configuration management tool.
4. The different configurations of the supporting software,
5. Special requirement for running machine-intensive tests
6. Appropriate number of licenses of all the software.

		Test Deliverables (Ref. book no.1, Chapter 15 Page 357) The deliverables include the following, 1. The test plan 2. Test case design specification 3. Test case 4. Test logs produced 5. Test summary report **Refer following Link for Deciding Test Approach-** • **www.unf.edu/~broggio/cen6017/ProjectDeliverables.Final.ppt** • **http://testlead.co.in/what-are-the-main-test-deliverables-for-software-testing/** **Testing Tasks(Size and Effort Estimation)** (Ref. book no.1, Chapter 15 Page 357 to 360) **Estimations happens broadly in three phases,** 1. Size estimation -Number of test cases -Number of test scenarios -Number of configuration to be tested 2. Effort estimation -Productivity data -Reuse opportunities -Robustness of processes 3. Schedule estimation
4	**4.2**	**Test Management** **Choice of Standards** (Ref. book no. 1 Chapter 15 Page 366 to 369) Internal Standard include, 1. Naming and storage conventions for test artifacts; 2. Document standards; 3. Test coding standards; 4. Test reporting standards **Test Infrastructure Management** (Ref. book no.1Chapter 15 Page 369) Testing requires a robust infrastructure to be planned upfront. This infrastructure

		is made up of three essential elements. 1. A test case database(TCDB) 2. A defect repository 3. Configuration management repository and tool A test case database captures all the relevant information about the test cases in an organization. Some of the entities and the attributes are given in following table A defect repository captures all the relevant details of defects reported for a product. The information that a defect repository includes is given in Table. TCDB, defect repository and SCM repository should complement each other and work together in an integrated fashion.
5	**4.2**	**Test People Management** (Ref. book no.1, Chapter 15 Page 372) • People management is an integral part of any project management. • People management also requires the ability to hire, motivate, and retain the right people. These skills are seldom formally taught. • Testing projects present several additional challenges. We believe that the success of a testing organization depends vitally on judicious people management skills **Refer following Link for Test People Management-** • http://www.softwaretestinghelp.com/how-to-build-a-successful-qa-team/ • http://www.methodsandtools.com/archive/archive.php?id=3
6	**4.2**	**Integrating with Product Release** (Ref. book no. 1 Chapter 15 Page 373) The success of a product depends on the effectiveness of integration of the development and testing activities. The schedules of testing have to be linked directly to product release. The following are some of the points to be decided for this planning- 1. Sync point between development and testing as to when different types of testing can commence. 2. Service level management between development and testing as to how long it would take for the testing team to complete the testing. 3. Consistent definitions of the various priorities and severities of the defects. 4. Communication mechanisms to the documentation group to ensure that the documentation is kept in sync with the product in terms of known defects, workarounds. **Refer following Link for Integrating with Product Release-** **1. http://www.techopedia.com/definition/24628/product-release-software** **2. http://en.wikipedia.org/wiki/Software_release_life_cycle**

7	4.3	**Test Process** • **Base Lining a Test Plan** (Ref. book no. 1 Chapter 15 Page 374) • A test combines all the points into a single document that acts as an anchor point for the entire testing project. • An organization normally arrives at a template that is to be used across the board. Each testing project puts together a test plan based on the template. • The test plan is reviewed by a designated set of component people in the organization. • After this, the test plan is baselined into the configuration management repository. • From then on, the baselined test plan becomes the basis for running the testing project. • In addition, periodically, any changes needed to the test plan templates are discussed among the different stake holders and this is kept current and applicable to the testing teams. **Refer following Link for Base Lining a Test Plan-** http://www.chambers.com.au/glossary/baseline.php
8	**4.3**	• **Test Case Specification** (Ref. book no.1 Chapter 15 Page 374) Using the test plan as the basis, the testing team designs ***test case specification,*** which then becomes the basis for preparing individual ***test cases***. Hence, a test case specification should clearly indentify, 1. The purpose of the test: This lists what features or part the test is intended for. 2. Items being tested, along with their version/release numbers as appropriate. 3. Environment that needs to be set up for running the test cases: This includes the hardware environment setup, supporting software environment setup, setup of the product under test. 4. Input data to be used for the test case: The choice of input data will be dependent on the test case itself and the technique followed in the test case. 5. Steps to be followed to execute the test: If automated testing is used, then, these steps ate translated to the scripting language of the tool. 6. The expected results that are considered to be "correct result". 7. A step to compare the actual result produced with the expected result: This step should do an "intelligent" comparison of the expected and actual results to highlight any discrepancies.

		8. Any relationship between this test and other test: These can be in the form of dependencies among the tests or the possibilities of reuse across the tests. **Refer following Link for Test Case Specification-** • **http://www.slideshare.net/gcrqtp/test-case-specification-template** • **http://www.onestoptesting.com/test-cases/test-specifications.asp** **Video refer from:** • **www.youtube.com/watch?v=KVdo7tPea_M** • **www.youtube.com/watch?v=XLyeY_QGm7s** **Update of Traceability** (Ref. book no.1 Chapter 15 Page 375) - Black Box Testing, a requirements traceability matrix ensures that the requirements make it through the subsequent life cycle phases and do not get orphaned mid-course. - The traceability matrix is a tool to validate that every requirement is tested. - The traceability matrix is created during the requirement gathering phase itself by filling up the unique identifier for each requirement. -This ensures that there is a two-way mapping between requirements and test cases.
9	**4.4**	**Test Reporting** **Recommending Product Release.** (Ref. book no.1 Chapter 15 Page 379) • Based on the test summary report, an organization can take a decision on whether to release the product or not. • Ideally, an organization would like to release a product with zero defects. • However, market pressures may cause the product to be released with the defects provided that the senior management is convinced that there is no major risk of customer dissatisfaction. • Such a decision should be taken by the senior manager after consultation with customer support team, development team, and testing team. **Matrix** A matrix is a concise organizer of simple tests, especially useful for function tests and domain tests. It groups test cases that are essentially the same. For example, for most input fields, you'll do a series of the same tests, checking how the field handles boundaries, unexpected characters, function keys, etc. To create a test matrix, you will have to: • Put the objects that you're testing on the rows. • Show the tests on the columns. • Check off the tests that you actually completed in the cells. **Refer following Link for Matrix-**

		• **http://www.softwaretestingtimes.com/2010/04/traceability-matrix-from-software.html** • **http://archanagaikwad.blogspot.in/**
10	**4.4**	**Executing Test Cases** (Ref. book no. 1,Chapter 15 Page 376) The prepared test cases have to be executed at the appropriate times during a project. As the test cases are executed during a test cycle, the defect repository is updated with, 1. Defect from the earlier test cycles that are fixed in the current build; 2. New defect that get uncovered in the current run of the tests. During test design and execution, the traceability matrix should be kept current. As and when tests get designed and executed successfully, the traceability matrix should be updated. **Refer following Link for Executing Test Cases-** **http://cs.anu.edu.au/student/comp8130/.../TestExecution_and_Reporting.ppt** **Collecting and Analyzing Metrics** (Ref. book no. 1 Chapter 15 Page 377) When test are executed, information about test execution gets collected in test logs and other files. The basic measurements from running the tests are then converted to meaningful metrics by the use of appropriate transformations and formulae. **Refer following Link for Collecting and Analyzing Metrics-** • **http://www.infosys.com/engineering-services/white-papers/Documents/comprehensive-metrics-model.pdf** • **http://www.softwaretestingtimes.com/2010/04/read-part-1-guide-to-effective-test.html** • **http://en.wikipedia.org/wiki/Software_metric**
11	**4.4**	**Preparing Test Summary Report** (Ref. book no. 1 Chapter 15 Page 377 and 378,379) At the completion of a test cycle, a test summary report is produced. This report gives insights to the senior management about the fitness of the product for release. There are two types of reports that are required: 1. The Incident Report 2. Test Cycle Report 3. Test Summary Report **A summary report should present;** 1. A summary of the activities carried out during the test cycle; 2. Variance of the activities carried out from the activities planned; 3. Summary of results should include tests that failed and severity of impact of defect;

		4. Comprehensive assessment and recommendation for release should include "Fit for release" assessment and Recommendation of release. **Refer following Link for Preparing Test Summary Report** • **www.softwaretestingmentor.com/test-deliverables/test-execution-reports/** • **www.softwaretestingstuff.com/2013/08/test-summary-report.html** • **https://cs.uwaterloo.ca/~palencar/cs447/lectures/Lecture20.ppt** **Video refer from:** • **https://www.youtube.com/watch?v=9GRtyvs5CTQ**
12		**Revision of Topic 4**

Topic 5: Defect Management.

Specific Objectives: (as given in curriculum)

1. Find, handle and report defect by using standard technique.
2. Understand the Defect Life Cycle.

Additional Special Objectives:

- Understand the concept of Defect and its classification.
- Understand the Defect Management Process.
- Apply the Techniques of finding and reporting defect practically**.**

Knowledge Category	Example(s) of Category	Teaching Methodology
FACT	Defect Introduction	Root Cause of Defect, Effect of a Defect Ref. book 1. Page no. 203,204; Use of following Video **https://www.youtube.com/watch?v=bp3MIYUk16w**
CONCEPT	Defect Classification	Requirement Defect, Design Defect, Coding Defect, Testing Defect Ref. book 1. Page no. 204,205;
PRINCIPLE	Defect Management Process, Defect Life Cycle	Demonstrate with following Videos **https://www.youtube.com/watch?v=3bCgGrEz8Kw**
PROCEDURE	Estimate Expected Impact of a Defect, Techniques for finding Defect	Ways to handle Risk Ref. book 1. Page no. 214 Minimize Risk impact or Probability Ref. book 1. Page no. 215
APPLICATION	Defect Template, Reporting a Defect	Refer following Link for Defect Template format- **http://www.softwaretestingmentor.com/defects/sample-defect-template/** Refer following Link for Defect Reporting format- **http://www.softwaretestinghelp.com/sample-bug-report/**

Learning Resources:

Books:

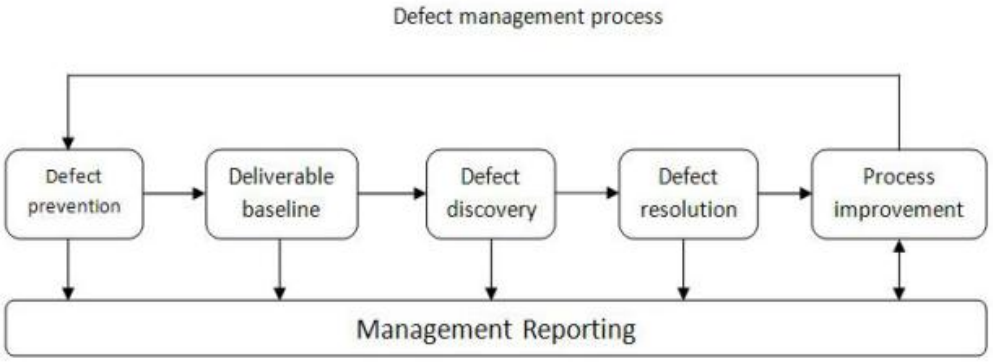
Title:	**1. Software Testing – Principles and practices**	Srinivas Desikan, Gopalaswamy Ramesh, Pearson
	2. Software Testing – Principles techniques and tools	M.G. Limaye, McGraw Hill
	3. Software Testing – Principles and	Naresh Chauhan, Oxford

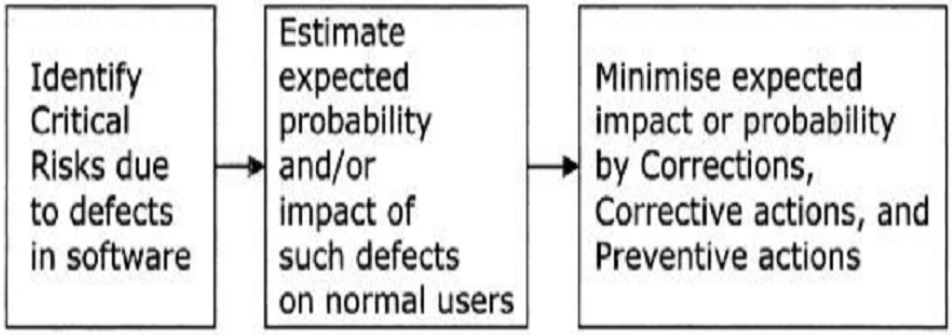
	practices	

Teaching Aids:
Black board, Chalk, Transparencies, Power point presentation slides(PPTs), Reference books, notes, LCD projector/OHP Projector

PPT with Sample:	1. **http://www.cs.toronto.edu/~penny/teaching/csc444-06f/L09-defect-tracking.ppt** 2. **www.msqaa.org/Best_Practices/.../DefectTrackingandManagement.ppt**
Website:	1. **http://www.softwaretestinghelp.com/** 2. **http://www.defectmanagement.com/defectmanagement/modelmain.htm** 3. **http://www.defectmanagement.com/defectmanagement/index.htm**

Lect. No.	Topic	Topics to be covered
1	**5.1**	• **Introduction of Defect Management** (Ref. book no. 2 Chapter 8 Page 203) ▪ Define Defect. ▪ Root Cause of Defect ▪ Effect of Defect. **Refer following Link for Introduction of Defect Management-** • **http://cdn.intechopen.com/pdfs-wm/9291.pdf** • **http://www.adminitrack.com/articles/defect-management-software-tools-system-plan.aspx** **Video refer from:** • **www.youtube.com/watch?v=M17OLBDtF4A** • **Defect Classification** (Ref. book no.2. Chapter 8 Page 204) Defect may be classified in Different ways under different schemes. 1. Requirement-related Defect may be further classified as- • Functional Defect • Interface Defect 2. Design Defect may be classified as follows- • Algorithm Defect • Module –Interface Defect • System –Interface Defect • User –Interface Defect 3. Coding Defect • Variable Declaration/Initialisation Defect • Database-Related Defect • Commenting/Documenting Defect 4. Testing Defect • Test-Design Defect

		• Test-Environment Defect • Test-Tool Defect **Refer PPT's from**: • http://cs.mwsu.edu/~stringfe/courseinfo/cs5443SQA/Lectures/Ch20.ppt
2	**5.1**	• **Defect Management Process(Approach)** (Ref. book no.2 Chapter 8 Page 206 and Page 212) Defect Management process must have the following characteristics: • Primary Goal of Defect Management Is to Prevent Defect • Defect Management Must Be Risk Driven • Defect Prevention Must Be an Integral Part of Development Process • Process of Defect Tracking Must be Automated As Maximum As Possible • Defect Integration Must Be Used for Process Improvement  • **Defect Management Process(Fixing and Root Cause Defect)** Defect Management process must include the appraisal of a defect finding process, software development process and the actions initiated to close the defects and strengthen the product/process associated with development, so that defects are not repeated again and again. It typically includes correction, corrective action and preventive action. It includes, • Defect Naming • Defect Resolution • Deliverable Base lining • Process Improvement • Management Reporting • **Defect Fixing** The process of risk analysis and defect prevention is refer from following figure:

		 Fig. 8.3 Defect-fixing process **Refer PPT's from:** • **http://www.authorstream.com/Presentation/tapas.pattanaik-360318-defect-management-process-clareta-labs-improvement-science-technology-ppt-powerpoint/** • **http://softwareenterprise.asu.edu/home/files/.../DefectManagement10.ppt**
3	**5.2**	**Defect Life Cycle-** (Ref. book no.2, Chapter 8 Page 207) Defect life cycle is a cycle which a defect goes through during its lifetime. It starts when defect is found and ends when a defect is closed, after ensuring it's not reproduced. Defect life cycle is related to the bug found during testing. The bug has different states in the Life Cycle. The Life cycle of the bug can be shown diagrammatically as follows:

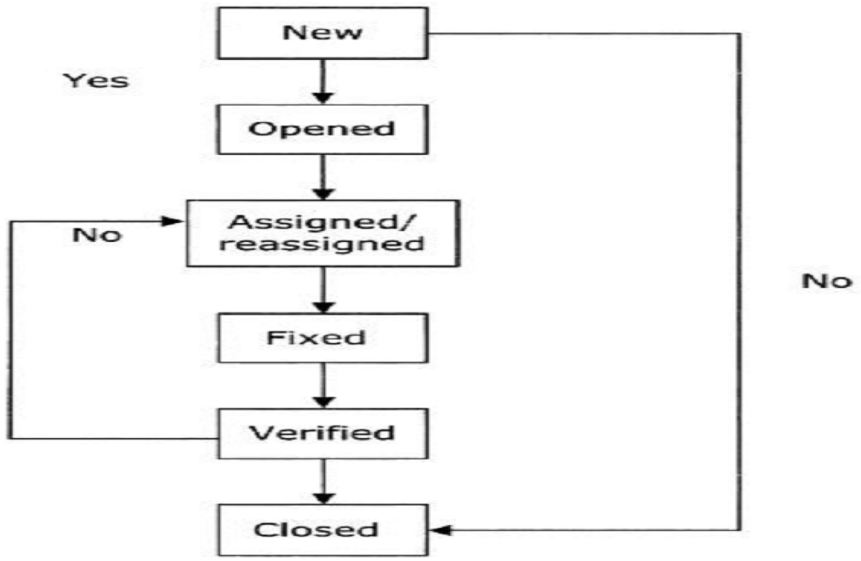


Fig. 8.2 Defect life cycle

1. **New:** When a defect is logged and posted for the first time. It's state is given as new.
2. **Assigned:** After the tester has posted the bug, the lead of the tester approves that the bug is genuine and he assigns the bug to corresponding developer and the developer team. It's state given as assigned.
3. **Open:** At this state the developer has started analyzing and working on the defect fix.
4. **Fixed:** When developer makes necessary code changes and verifies the changes then he/she can make bug status as 'Fixed' and the bug is passed to testing team.
5. **Pending retest:** After fixing the defect the developer has given that particular code for retesting to the tester. Here the testing is pending on the testers end. Hence its status is pending retest.
6. **Retest:** At this stage the tester do the retesting of the changed code which developer has given to him to check whether the defect got fixed or not.
7. **Verified:** The tester tests the bug again after it got fixed by the developer. If the bug is not present in the software, he approves that the bug is fixed and changes the status to "verified".
8. **Reopen:** If the bug still exists even after the bug is fixed by the developer, the tester changes the status to "reopened". The bug goes through the life cycle once again.
9. **Closed:** Once the bug is fixed, it is tested by the tester. If the tester feels that the bug no longer exists in the software, he changes the status of the bug to "closed". This state means that the bug is fixed, tested and approved.
10. **Duplicate:** If the bug is repeated twice or the two bugs mention the same concept of the bug, then one bug status is changed to "duplicate**".**
11. **Rejected:** If the developer feels that the bug is not genuine, he rejects the bug.

Then the state of the bug is changed to "rejected".

12. **Deferred:** The bug, changed to deferred state means the bug is expected to be fixed in next releases. The reasons for changing the bug to this state have many factors. Some of them are priority of the bug may be low, lack of time for the release or the bug may not have major effect on the software.
13. **Not a bug:** The state given as "Not a bug" if there is no change in the functionality of the application. For an example: If customer asks for some change in the look and field of the application like change of color of some text then it is not a bug but just some change in the looks of the application.

Refer Video from:-

- **https://www.youtube.com/watch?v=SwRzLondC_Y**
- **www.youtube.com/watch?v=_PejNa-v5tY**

Refer following Link for Defect Life Cycle-

- **http://softwaretestingfundamentals.com/defect-life-cycle/**
- **http://www.tutorialspoint.com/software_testing_dictionary/defect_life_cycle.htm**

Defect Template-

(Ref. book no.2 Chapter 8 Page 209)

A defect report documents an anomaly discovered during testing. It includes all the information needed to reproduce the problem, including the author, release/build number, open/close dates, problem area, problem description, test environment, defect type, how it was detected, who detected it, priority, severity, status, etc.

After uncovering a defect (bug), testers generate a formal defect report. The purpose of a defect report is to state the problem as clearly as possible so that developers can replicate the defect easily and fix it.

DEFECT REPORT TEMPLATE

In most companies, a defect reporting tool is used and the elements of a report can vary. However, in general, a defect report can consist of the following elements.

ID	Unique identifier given to the defect. (Usually Automated)
Project	Project name.
Product	Product name.
Release Version	Release version of the product. (e.g. 1.2.3)
Module	Specific module of the product where the defect was detected.
Detected Build Version	Build version of the product where the defect was detected (e.g. 1.2.3.5)
Summary	Summary of the defect. Keep this clear and concise.
Description	Detailed description of the defect. Describe as much as possible but without repeating anything or using

	complex words. Keep it simple but comprehensive.
Steps to Replicate	Step by step description of the way to reproduce the defect. Number the steps.
Actual Result	The actual result you received when you followed the steps.
Expected Results	The expected results.
Attachments	Attach any additional information like screenshots and logs.
Remarks	Any additional comments on the defect.
Defect Severity	Severity of the Defect. (See Defect Severity)
Defect Priority	Priority of the Defect. (See Defect Priority)
Reported By	The name of the person who reported the defect.
Assigned To	The name of the person that is assigned to analyze/fix the defect.
Status	The status of the defect. (See Defect Life Cycle)
Fixed Build Version	Build version of the product where the defect was fixed (e.g. 1.2.3.9)

REPORTING DEFECTS EFFECTIVELY

It is essential that you report defects effectively so that time and effort is not unnecessarily wasted in trying to understand and reproduce the defect. Here are some guidelines:

- Be specific
- Be detailed
- Be objective
- Reproduce the defect
- Review the report

Table 8.1 Defect phases and corresponding status

Defect phases	Typical status
New defect found	Open
Defect assigned to concerned person (generally module lead, project lead, team lead etc)	Assigned <Name of person>
Defect is duplicate	Duplicate
Defect is as per requirements	Not a defect
Defect cannot be reproduced	Cannot be reproduced
Defect assigned to person who will be fixing it	Assigned <Name of person>
Defect fixed by the identified person	Fixed
Defect verified by module lead, team lead, project lead etc	Verified
Defect after fixing assigned to tester	Assigned <Name of person>
Defect found to be fixed	Closed
Defect found to be open	Reopen

Refer following Link for Defect Template –

		• **www.testinggeek.com/defect-report-template** • **http://www.testuff.com/help/defect-template/** **Refer Video from:-** • **www.youtube.com/watch?v=Qn9uzwrM9CY** • **https://www.youtube.com/watch?v=jUANZOGS7bg**

4	**5.3**	• **Estimate Expected Impact of a Defect-** (Ref. book no.2 Chapter 8 Page 214) ✓ Ways To Handle Risk - Accept the Risk As It Is - Bypassing the Risk ✓ Minimize Risk Impact or Probability ✓ Minimization of problem due to risk happens in the following manner. • Eliminate Risk • Mitigation of Risk • Detection Ability Improvement • Contingency Planning ✓ **Techniques for Finding Defect-** (Ref. book no.2 Chapter 8 Page 216) Defects are found either by preplanned activities specifically intended to uncover defects (e.g., quality control activities such as inspections, testing, etc.) or by accident (e.g., users in production).Techniques to find defects can be divided into three categories: • ***Static techniques***: Testing that is done without physically executing a program or system. A code review is an example of a static testing technique. • ***Dynamic techniques***: Testing in which system components are physically executed to identify defects. Execution of test cases is an example of a dynamic testing technique. • ***Operational techniques***: An operational system produces a deliverable containing a defect found by users, customers, or control personnel -- i.e., the defect is found as a result of a failure. While it is beyond the scope of this study to compare and contrast the various static, dynamic, and operational techniques, the research did arrive at the following conclusions: • Both static and dynamic techniques are required for an effective defect management program. In each category, the more formally the techniques were integrated into the development process, the more effective they were. • Since static techniques will generally find defects earlier in the process, they are more efficient at finding defects.
5		• **Reporting a Defect-** (Ref. book no.2, Chapter 8 Page 216 and Page 217) Defect reports communicate issues to developers and other stakeholders. When a defect is initially uncovered, it may be unclear whether it is a defect, an

undocumented requirements change, a user error, or a misunderstanding. Some may resist calling something a “defect” because that implies "bad work" and might not reflect well on the team. Others may resist calling something a “requirements change" because that implies that requirements or design specifications were missed and changes may cost more money.

Some organizations try to defuse this issue by changing the name to something less inflammatory, like “incident" or "issue" report. From a defect management perspective, what they are called is not an important issue. What is important is that the defect be quickly brought to the developers' attention and formally controlled.

A good defect report should:

1. Give sufficient and high-quality information to reproduce and fix the defect;
2. Be well written; and
3. Enable stakeholders to make wise decisions about which defects to fix.

- **Issues Should Be Reported and Acknowledged**

Reporting a defect about someone else's work can be difficult. Developers can feel frustrated and bitter toward testers when defects are found. When there is a debate on whether to report or fix a defect, both parties can feel aggravated and apprehensive. Care should be taken that all parties (to include the rest of the team – requirements, project management) are non-judgmental and unemotional.

Most importantly, Quality Control and Development stakeholders must remember the ultimate project goal is to maximize the value of the application, and it is up to management/other stakeholders to decide that defects are handled correctly and given the right priority. This will enhance teamwork between Quality Control and Development.

- **Defect Reports Should Clearly State the Problem-**

A Developer should be able to easily replicate and fix a defect that has been identified. Don’t waste the developer’s time by including extraneous information. At Segue, we use **Test Track Pro** to report and track issues. Test Track Pro has a defect report template that ensures the correct level of detail is captured. Using a defect template becomes especially useful when documenting defects found in complex systems.

Here at Segue, we follow this process to support multiple Air Force information systems with a great number of specific acronyms and business cases. When we find defects in such systems, we often include screenshots and videos of these defects to reduce ambiguity and confusion.

This enables the developers to quickly reproduce and fix bugs. The information included needs to be specific and should include a step-by-step account of the actions to reproduce the defect to avoid miscommunication. Do not skip seemingly “intuitive” steps or assume the developer knows and remembers the steps for every process. For example, it may not be sufficient enough to say “select the value,” if selecting the value can be accomplished by double clicking or checking a box or using a drop down menu.

Also, do not use your defect report to make subjective criticisms; your report is to

help stakeholders make an informed decision and that decision may be to NOT fix the defect. To ensure pertinent information is documented, the following fields, which are a part of our defect template, should be required in your defect report.

• ID	• Component\Feature
• Description	• Expected Results
• Product	• Summary
• Steps to Replicate	• Reported By
• Release Version	

- ***Defect Resolution***

Establishing a defect resolution process is great for the inevitable situation in which testers, requirements analysts, and developers are in dispute regarding a defect. For example, when a stakeholder reports a defect, but the developers do not believe it to be a defect or important issue, a quick-resolution process would be beneficial. While many approaches can address this situation, the two most effective are:

1. ***Arbitration by the software owner***: the customer using the software determines whether or not the problem is a defect.
2. ***Arbitration by a software development manager***: a senior manager of the software development department will be selected to resolve the dispute.

Defect reports are among the most important deliverables to come out of testing. They have a direct impact on the quality of the product, so a tester should take the time to create a great defect report. CemKaner, a Professor of Software Engineering, said "The best tester isn't the one who finds the most bugs or who embarrasses the most programmers.

The best tester is one who gets most bugs fixed." As you can see, it is worth the effort to learn how to write effective defect reports include following important points in defect reporting,

- Give Complete Record of Discrepancies
- Defect Report Forms the Base for Quality Measurement (for the software As well as Process)

Include Defect Management important stages,

- Correct the Defect
- Report Status of System
- Gather Statistics to Predict Future
- Process Improvement

Websites:

- **http://www.softwaretestinghelp.com/how-to-write-good-bug-report/**

Refer Video from:-

- **www.youtube.com/watch?v=zoohdktgGwY**
- **www.youtube.com/watch?v=y1ToonseeBU**

6		**Revision of Topic 5**

Topic 6: Testing Tools and Measurements

Specific Objectives: (as given in curriculum)

1. Understand the shortcomings of manual testing.
2. Understand the use of automated test tools.

Additional Special Objectives:

- Understand the benefits of Automation Testing.
- Understand the Tools used in automation testing and use appropriate automation testing tool.

Knowledge Category	Example(s) of Category	Teaching Methodology
FACT	Limitations of Manual Testing and Need for Automated Testing Tools	Ref. book no.3 Chapter 15 Page 430; **Refer following Link for Limitation of Manual Testing -** **www.gcreddy.com/2014/01/disadvantages-of-manual-testing.html**
CONCEPT	Advantages and Disadvantages of Using Tools	Ref. book no.2**,** Chapter 13 Page 320; Refer following Link - **http://en.wikipedia.org/wiki/Test_management** **http://www.softwaretestinghelp.com/test-data-management-techniques/**
PRINCIPLE	Features of Test Tool; What are Metrics and Measurement	Features of Test Tool Types of Metrics, Ref. book 1. Page no.427 Project Metrics, Ref. book 1. Page no.428 Progress and Productivity Metrics; Ref. book 1. Page no. 434,448
PROCEDURE	Selecting Testing Tool	Ref. book no.1 Chapter 16 Page 411 Refer following Link for Selecting Testing Tool– 1. **http://www.softwaretestingstandard.org/part2.php** 2. **http://www.slideshare.net/tokarthik/Test-Process**
APPLICATION	When to use Automated Testing Tools and Testing using Automated Tools	Ref. book no.2 Chapter 13 Page 320,321 Refer following Link - • **http://www.softwaretestinghelp.com/how-to-write-test-plan-document-software-testing-training-day3/**

<table>
<tr><td colspan="4">Learning Resources:</td></tr>
<tr><td colspan="4">Books:</td></tr>
<tr><td rowspan="2">Title:</td><td colspan="2">1. Software Testing – Principles and practices</td><td>Srinivas Desikan, Gopalaswamy Ramesh, Pearson</td></tr>
<tr><td colspan="2">2. Software Testing – Principles techniques and tools</td><td>M.G. Limaye, McGraw Hill</td></tr>
<tr><td></td><td colspan="2">3. Software Testing – Principles and practices</td><td>Naresh Chauhan, Oxford</td></tr>
<tr><td colspan="4">Teaching Aids:
Black board, Chalk, Transparencies, Power point presentation slides(PPTs), Reference books, notes, LCD projector/OHP Projector</td></tr>
<tr><td colspan="2">Video Lectures:</td><td colspan="2">PPT with Sample: -
1. www.youtube.com/watch?v=XLyeY_QGm7s
2. www.youtube.com/watch?v=f3U6V_5HEhk
3. www.youtube.com/watch?v=_TzrG53WF0g</td></tr>
<tr><td colspan="2">Website:</td><td colspan="2">1. www.slideshare.net/Programeter/measures-and-metrics-5266148
2. http://www.aptest.com/resources.html
3. http://www.testinggeek.com/software-testing-measurement
4. www.cs.uku.fi/tutkimus/Teho/SoftwareTestingTools.pdf</td></tr>
<tr><td>Lect. No.</td><td>Topic</td><td colspan="2">Topics to be covered</td></tr>
<tr><td>1</td><td>6.1</td><td colspan="2">Limitations of Manual Testing
• Manual testing is not reliable. Using this method test execution is not accurate all the time.
• To execute the test cases first time using manual testing will be very much useful. But it is not sure that it will catch the regression defects under frequently changing requirements.
• Manual testing will be useful when the test case only needs to run once or twice.
• To execute the test cases every time tester requires the same amount of time.
• Using manual testing, testing on different machine with different OS platform combination is not possible, concurrently. To execute such task different testers are required.
• It does not involve in programming task to fetch hidden information.
• Manual testing is slower than automation. Running tests manually can be very time consuming.
Refer following Link for Limitations of Manual Testing -
• http://www.softwaretestingclass.com/automation-testing-vs-manual-testing/

Need for Automated Testing Tools
(Ref. book no.3 Chapter 15 Page 430)</td></tr>
</table>

Automation means taking your manual tests and automating them using a tool and language of your choice. As a tester, I have also looked into automated testing and to make things clearer for us, I have drawn up a list of points as to why I feel projects will benefit from automated testing:

Save Time –

This is one of the biggest benefits for me, especially when it comes to regression testing. As we all know, regression testing is the retesting of the application when new features have been introduced or a change is made to an existing feature that has been previously tested (the change can be as a result of change request, defect fix, refactoring).

The aim of regression testing is to ensure the application still works as expected and in order for us to verify this, we need to run all test scripts associated with the change. There is a risk here, that due to time constraints we may not run all tests associated to the change which may result in undiscovered defects.

These issues can be overcome by setting up our automated tests to run over night or after each deployment – this creates time for the tester to perform exploratory testing, concentrate on areas which cannot be automated and concentrate on other tasks.

Speed –

As automated tests are run by tools, these are run much faster than human users which add to the first benefit of saving time.

Repeatability –

The same tests can be re-run in exactly the same manner eliminating the risk of human errors such as testers forgetting their exact actions, intentionally omitting steps from the test scripts, missing out steps from the test script, all of which can result in either defects not being identified or the reporting of invalid bugs (which can again, be time consuming for both developers and testers to reproduce)

Maintenance of the test suite –

New functionality has been introduced or existing features have been changed in the way they work and the test cases are no longer up to date as the tester(s) has had no time to go back and update the test scripts. When tests are automated and run after each build, those that are out of date will fail hence, forcing the tester to go back and fix the test script – this process ensures the test scripts are kept up to date and quality of the software is maintained

Reusable –

The automated tests can be reused on different versions of the software, even if the interface changes.

Increase Coverage –

Testers can create a test suite with tests to cover every feature within the application.

Cost Reduction –

This can be of benefit when the numbers of resources required for regression testing are reduced.

		In a nutshell, automation helps to save time by reducing the time taken to run tests; Increases the quality of the software and testing process through reliability, repeatability and comprehensiveness of the test suite; Utilizing manpower more effectively by applying skills and time where they are needed most and increasing test coverage. Of course, automating tests does not mean we are eliminating the tester's role. There will always be a place for a human tester within a project team, as not every test associated with a feature can be automated and not every project is suitable for automated testing. As a tester, automation is about making our lives easier, by using it to overcome problems such as time and testing more efficiently to ensure quality is maintained not just within the application being developed but also within the testing process. **Refer following Link for Scope Management-** • **http://red-badger.com/blog/2013/02/01/benefits-of-automated-testing/** • **http://www.ranorex.com/why-test-automation.html**
	6.2	**Features of Test Tool** **Guideline for Static and Dynamic Testing Tool** • **Static testing** : Static testing is a form of software testing where the software isn't actually used. This is in contrast to dynamic testing. It is generally not detailed testing, but checks mainly for the sanity of the code, algorithm, or document. It is primarily checking of the code and/or manually reviewing the code or document to find errors. This type of testing can be used by the developer who wrote the code, in isolation. Code reviews, inspections and Software walkthroughs are also used. From the black box testing point of view, static testing involves reviewing requirements and specifications. This is done with an eye toward completeness or appropriateness for the task at hand. This is the **verification** portion of Verification and Validation. Even static testing can be automated. A static testing test suite consists of programs to be analyzed by an interpreter or a compiler that asserts the programs syntactic validity. Bugs discovered at this stage of development are less expensive to fix than later in the development cycle. The people involved in static testing are application developers and testers. • **Static analysis tools:** Static analysis tools are generally used by developers as part of the development and component testing process. The key aspect is that the code (or other artefact) is not executed or run but the tool itself is executed, and the source code we are interested in is the input data to the tool. These tools are **mostly used by developers.** Static analysis tools are an extension of compiler technology – in fact some compilers do offer static analysis features. It is worth checking what is available from existing compilers or development environments before looking

at purchasing a more sophisticated static analysis tool.

Other than software code, static analysis can also be carried out on things like, static analysis of requirements or static analysis of websites (for example, to assess for proper use of accessibility tags or the following of HTML standards).

Static analysis tools for code can help the developers to understand the structure of the code, and can also be used to enforce coding standards.

Features or characteristics of static analysis tools are:

- To calculate metrics such as cyclomatic complexity or nesting levels (which can help to identify where more testing may be needed due to increased risk).
- To enforce coding standards.
- To analyze structures and dependencies.
- Help in code understanding.
- To identify anomalies or defects in the code.

Static Test Tools:

These tools do not involve actual input and output. Rather, they take a symbolic approach to testing, i.e. they do not test the actual execution of the software. These tools include the following: ,

1) **Flow analyzers:** They ensure consistency in data flow from input to output.
2) **Path tests:** They find unused code and code with contradictions.
3) **Coverage analyzers:** It ensures that all logic paths are tested.
4) **Interface analyzers:** It examines the effects of passing variables and data between modules.

Refer following Link for Guideline for Static Testing Tool-

- **http://istqbexamcertification.com/what-is-static-analysis-tools-in-software-testing/**
- **http://www.tutorialspoint.com/software_testing_dictionary/static_testing.htm**
- **http://en.wikipedia.org/wiki/Static_testing**
- **http://www.softwaretestinggenius.com/know-the-static-versus-dynamic-testing-tools**

Video refer from:

- **www.youtube.com/watch?v=Kktes8O_ywk**
- **www.youtube.com/watch?v=e1W_CYBbDhc**

Dynamic Testing:

Dynamic testing (or dynamic analysis) is a term used in software engineering to describe the testing of the dynamic behavior of code. That is, dynamic analysis refers to the examination of the physical response from the system to variables that are not constant and change with time. In dynamic testing the software must actually be compiled and run. It involves working with the software, giving input values and checking if the output is as expected by

executing specific test cases which can be done manually or with the use of an automated process. This is in contrast to static testing. Unit tests, integration tests, system tests and acceptance tests utilize dynamic testing.

Dynamic analysis tools

Dynamic analysis tools are 'dynamic' because they require the code to be in a running state. They are 'analysis' rather than 'testing' tools because they analyse what is happening 'behind the scenes' that is in the code while the software is running (whether being executed with test cases or being used in operation).

Let us take an example of a car to understand it in a better way. If you go to a showroom of a car to buy it, you might sit in the car to see if is comfortable and see what sound the doors make – this would be static analysis because the car is not being driven. If you take a test drive, then you would check that how the car performs when it is in the running mode e.g. the car turns right when you turn the steering wheel clockwise or when you press the break then how the car will respond and can also check the oil pressure or the brake fluid, this would be dynamic analysis, it can only be done while the engine is running.

Features or characteristics of dynamic analysis tools are as follows:

- To detect memory leaks;
- To identify pointer arithmetic errors such as null pointers;
- To identify time dependencies.

Eventually when your computer's response time gets slower and slower, but it get improved after re-booting, this may be because of the 'memory leak', where the programs do not correctly release blocks of memory back to the operating system. Sooner or later the system will run out of memory completely and stop. Hence, rebooting restores all of the memory that was lost, so the performance of the system is now restored to its normal state.

These tools would typically be used by developers in component testing and component integration testing, e.g. when testing middleware, when testing security or when looking for robustness defects.

Another form of dynamic analysis for websites is to check whether each link does actually link to something else (this type of tool may be called a 'web spider'). The tool does not know if you have linked to the correct page, but at least it can find dead links, which may be helpful.

Dynamic Test Tools:

These tools test the software system with 'live' data. Dynamic test tools include the following

1. **Test driver:** It inputs data into a module-under-test (MUT).
2. **Test beds:** It simultaneously displays source code along with the program under execution.
3. **Emulators:** The response facilities are used to emulate parts of the system

		not yet developed. 4. **Mutation analysers:** The errors are deliberately 'fed' into the code in order to test fault tolerance of the system. **Refer following Link for Guideline for Dynamic Testing Tool-** • **http://en.wikipedia.org/wiki/Dynamic_testing** • **http://istqbexamcertification.com/what-is-dynamic-analysis-tools-in-software-testing/** • **http://www.softwaretestinggenius.com/know-the-static-versus-dynamic-testing-tool** **Video refer from:** • **www.youtube.com/channel/UC2wP2qn8ScHQ4D9efHRqUhw**
2	**6.3**	**Advantages and Disadvantages of Using Tools** (Ref. book no.2**,** Chapter 13 Page 320) There are many benefits that can be gained by using tools to support testing. They are: ▪ **Reduction of repetitive work:** Repetitive work is very boring if it is done manually. People tend to make mistakes when doing the same task over and over. Examples of this type of repetitive work include running regression tests, entering the same test data again and again (can be done by a test execution tool), checking against coding standards (which can be done by a static analysis tool) or creating a specific test database (which can be done by a test data preparation tool). ▪ **Greater consistency and repeatability:** People have tendency to do the same task in a slightly different way even when they think they are repeating something exactly. A tool will exactly reproduce what it did before, so each time it is run the result is consistent. ▪ **Objective assessment:** If a person calculates a value from the software or incident reports, by mistake they may omit something, or their own one-sided preconceived judgments or convictions may lead them to interpret that data incorrectly. Using a tool means that subjective preconceived notion is removed and the assessment is more repeatable and consistently calculated. Examples include assessing the cyclomatic complexity or nesting levels of a component (which can be done by a static analysis tool), coverage (coverage measurement tool), system behaviour (monitoring tools) and incident statistics (test management tool). ▪ **Ease of access to information about tests or testing:** Information presented visually is much easier for the human mind to understand and interpret. For example, a chart or graph is a better way to show information than a long list of numbers – this is why charts and graphs in spread sheets are so useful. Special purpose tools give these features directly for the information they process. Examples include statistics and graphs about

test progress (test execution or test management tool), incident rates (incident management or test management tool) and performance (performance testing tool).

Although there are many benefits that can be achieved by using tools to support testing activities, but there are also many risks that are associated with it when tool support for testing is introduced and used.

Risks include:

- **Unrealistic expectations from the tool:**
  Unrealistic expectations may be one of the greatest risks to success with tools. The tools are just software and we all know that there are many problems associated with any kind of software. It is very important to have clear and realistic objectives for what the tool can do.
- **People often make mistakes by underestimating the time, cost and effort for the initial introduction of a tool:**
  Introducing something new into an organization is hardly straightforward. Once you purchase a tool, you want to have a number of people being able to use the tool in a way that will be beneficial. There will be some technical issues to overcome, but there will also be resistance from other people – both need to be handled in such a way that the tool will be of great success.
- **People frequently miscalculate the time and effort needed to achieve significant and continuing benefits from the tool:**
  Mostly in the initial phase when the tool is new to the people, they miscalculate the time and effort needed to achieve significant and continuing benefits from the tool. Just think back to the last time you tried something new for the very first time (learning to drive, riding a bike, skiing). Your first attempts were unlikely to be very good but with more experience and practice you became much better. Using a testing tool for the first time will not be your best use of the tool either. It takes time to develop ways of using the tool in order to achieve what is expected.
- **Mostly people underestimate the effort required to maintain the test assets generated by the tool:**
  Generally people underestimate the effort required to maintain the test assets generated by the tool. Because of the insufficient planning for maintenance of the assets that the tool produces there are chances that the tool might end up as 'shelf-ware', along with the previously listed risks.
- **People depend on the tool a lot (over-reliance on the tool):**
  Since there are many benefits that can be gained by using tools to support testing like reduction of repetitive work, greater consistency and repeatability, etc. people started to depend on the tool a lot. But the tools are just a software they can do only what they have been designed to do (at least a good quality tool can), but they cannot do everything. A tool can definitely help, but it cannot replace the intelligence needed to know how best to use it, and how to evaluate current and future uses of the tool. For example, a test execution tool

		does not replace the need for good test design and should not be used for every test – some tests are still better executed manually. A test that takes a very long time to automate and will not be run very often is better done manually. **Refer following Link for Advantages and Disadvantages of Using Tools -** • **http://istqbexamcertification.com/what-are-the-advantages-or-benefits-of-using-testing-tools/** • **http://istqbexamcertification.com/what-are-the-risks-or-disadvantages-of-using-the-testing-tools/**
3	**6.4**	**Selecting a Testing Tool** (Ref. book no.1 Chapter 16 Page 411) Selecting the test tool is an important aspect of test automation for several reasons as given below: 1. Free tools are *not well supported and get phased out* soon. 2. Developing in-house tools *takes time*. Even though in-house tools can be less expensive and can meet need better, they are often developed by the personal interest shown by a few engineers. 3. Test tools sold by vendors are *expensive*. 4. Test tools require strong *training*. Test automation cannot be successful unless the people using tools are properly trained. 5. Test tools generally *do not meet all the requirements* for automation. Since tools are meant to be generic, they may not fully satisfy the needs of a particular customer. 6. Not all test tools run on all platforms. To amortize the costs of automation, the tools and the automated tests should be reusable on all the platforms on which the product under test runs. **Criteria for Selecting Test Tools:** The Categories for selecting Test Tools are, 1. Meeting requirements; 2. Technology expectations; 3. Training/skills; 4. Management aspects. • **Meeting requirements-** There are plenty of tools available in the market but rarely do they meet all the requirements of a given product or a given organization. Evaluating different tools for different requirements involve significant effort, money, and time. Given of the plethora of choice available, huge delay is involved in selecting and implementing test tools. • **Technology expectations-** Test tools in general may not allow test developers to extends/modify the functionality of the framework. So extending the functionality requires going back to the tool vendor and involves additional cost and effort. A good number of test tools require their libraries to be linked with product binaries.

- **Training/skills-**
  While test tools require plenty of training, very few vendors provide the training to the required level. Organization level training is needed to deploy the test tools, as the user of the test suite are not only the test team but also the development team and other areas like configuration management.
- **Management aspects-**
  A test tool increases the system requirement and requires the hardware and software to be upgraded. This increases the cost of the already- expensive test tool.

Table – Issues in selecting a testing tools.

Meeting Requirements	**Technology expectations**	**Training/ Skills**	**Management aspects**
Checking whether the tools meet requirements, involves effort and money	Extending the test tool is difficult	Lack of trainer for test tools	Test tool require system upgrades
Test tools are not fully compatible with product	Require instrumental code to be removed for certain tests	Test tools requires people to learn new language/scripts	Migration to other test tools difficult
Test tools are not tested with the same seriousness as products for new requirements	Test tools are not cross platform		Deploying tool requires huge planning and effort
Difficult to isolate problems of product and test suite; changes in product causes test suite to be changed			

Steps for Tool Selection and Deployment:

The steps in Tool Selection and Deployment are,

1. Identify your Test suite requirements among the generic requirements discussed. Add other requirement (if any).
2. Make sure experiences discussed in previous sections are taken care of.
3. Collect the experiences of other organizations which used similar test tool.
4. Keep a checklist of questions to be asked to the vendors on cost/effort/support.
5. Identify list of tools that meet the above requirement.
6. Evaluate and shortlist one/set of tools and train all test developers on the tool.
7. Deploy the tool across test teams after training all potential users of the tool.

Refer following Link for Selecting a Testing Tool -

- **www.testhouse.net/test-tool-slection**
- **http://istqbexamcertification.com/what-are-the-important-factors-for-the-software-testing-tool-selection/**

Refer Video from-

		• **www.youtube.com/watch?v=c4u6tUBfWUI** • **http://www.powershow.com/view/3b9e2e-NDU0N/Tips_for_Success_when_Selecting_Testing_Tools_powerpoint_ppt_presentation**
4	**6.5**	**Automated Test Tools** **When to Use Automated Test Tools-** (Ref. book no.2 Chapter 13 Page 320,321) Automated tests are a very useful and impressive tool used to make testing more efficient. However, automated tests are not suited to all projects – this may be due to lack of time available, or due to technical limitations. Automated tests take time to create. Depending on the testers, it can take 3-10 times the amount of time to create an automated test as it takes to run the same test manually. Therefore, automated tests will only start to be valuable when they are run more than 3-10 times. Automated tests are suitable for the following purposes: - Regression testing for a stable system that will be run on a regular basis - Fast data creation in test systems where the database must be wiped on a regular basis Automated tests are NOT suitable for the following purposes: - Testing new functionality – this should be done manually before automated tests are created - Regression testing systems that are expected to have significant user interface changes. Large changes to the user interface require a lot of maintenance for automated tests. When automating tests, it is wise to only automate as many tests as your team can easily maintain. If some tests are becoming difficult to maintain, it may be worth considering retiring those tests. Above all, remember that automated tests will never detect as many bugs as a human tester executing the same steps. This is because the human tester's eyes can pick up many things, whereas the test will only notice what it is programmed to notice. **Refer following Link for When to Use Automated Test Tools-** • **http://trishkhoo.com/2009/03/when-to-use-automated-tests/** • **http://www.softwaretestinghelp.com/10-tips-you-should-read-before-automating-your-testing-work/** **Testing Using Automated Tools-** (Ref. book no.2, Chapter 13 Page 321) In software testing, test automation is the use of special software (separate from the software being tested) to control the

execution of tests and the comparison of actual outcomes with predicted outcomes. Test automation can automate some repetitive but necessary tasks in a formalized testing process already in place, or add additional testing that would be difficult to perform manually.

Some software testing tasks, such as extensive low-level interface regression testing, can be laborious and time consuming to do manually. In addition, a manual approach might not always be effective in finding certain classes of defects. Test automation offers a possibility to perform these types of testing effectively. Once automated tests have been developed, they can be run quickly and repeatedly.

Many times, this can be a cost-effective method for regression testing of software products that have a long maintenance life. Even minor patches over the lifetime of the application can cause existing features to break which were working at an earlier point in time.

There are many approaches to test automation, however below are the general approaches used widely:

- **Code-driven testing**. The public (usually) interfaces to classes, modules or libraries are tested with a variety of input arguments to validate that the results that are returned are correct.
- **Graphical user interface testing**. A testing framework generates user interface events such as keystrokes and mouse clicks, and observes the changes that result in the user interface, to validate that the observable behaviour of the program is correct.
- **API driven testing**. A testing framework that uses a programming interface to the application to validate the behaviour under test. Typically API driven testing bypasses application user interface altogether.

Test automation is a process of writing a computer program to do testing that would otherwise need to be done manually. Once tests have been automated, they can be run quickly and repeatedly. This is often the most cost effective method for software products that have a long maintenance life, because even minor patches over the lifetime of the application can potentially cause working functionality (at an earlier point in time) to break.

1. **QTP**

HP-QuickTest Professional software provides functional and regression test automation for software applications and environments. HP QuickTest Professional supports keyword and scripting interfaces and features a graphical user interface. Its features are: a cascaded optimization system, the industry's deepest and broadest insight into IT-controlled assets, and a secure, comprehensive, operational environment for a hybrid world, enhanced expert view, business process testing, screen recorder etc.

2. **Selenium**

Selenium is a portable software testing framework for web applications. Selenium provides a record/playback tool for authoring tests without learning a test scripting language. It includes features like record and playback, intelligent field selection, Xpath as needed, auto complete for all common selenium commands, walk through tests, debug and set breakpoints, ruby scripts, or other formats, support for selenium user-extensions file, option to automatically assert the title of every page etc.

3. **VISUAL STUDIO TEST PROFESSIONAL**

Visual Studio Test Professional is an integrated testing toolset developed by Microsoft that delivers a complete plan-test-track workflow for in-context collaboration between testers and developers, in order to increase testers' visibility to the overall project. Its features are file actionable bugs, manual testing, re-use manual test recordings, team foundation server, application lifecycle management etc. Whilst rich in features, it is an observation that testing professionals may get overwhelmed and intimidated due to abundance of menu items in the software that have no relevance to them.

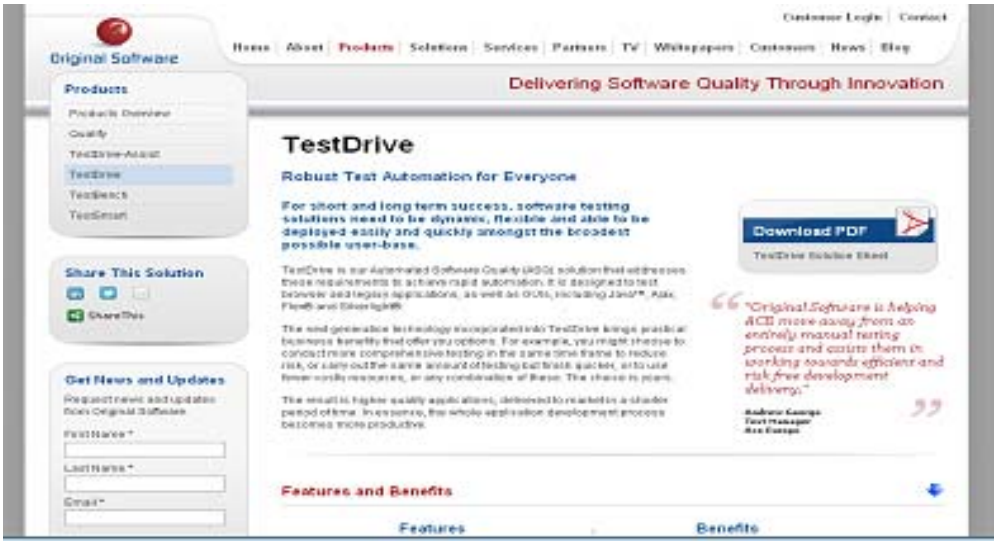
4. RATIONAL FUNCTIONAL TESTER

Rational Functional Tester is an automated functional testing and regression testing tool. Provides testers with automated testing capabilities for functional testing, regression testing, and GUI testing and data-driven testing. It's features are: simplify test creation and visualization with storyboard testing, provides lifecycle traceability, validate dynamic data with dynamic data validation wizard, streamline automation with keyword testing, proxy SDK, test script version control for parallel development, etc.

5. **TESTCOMPLETE**

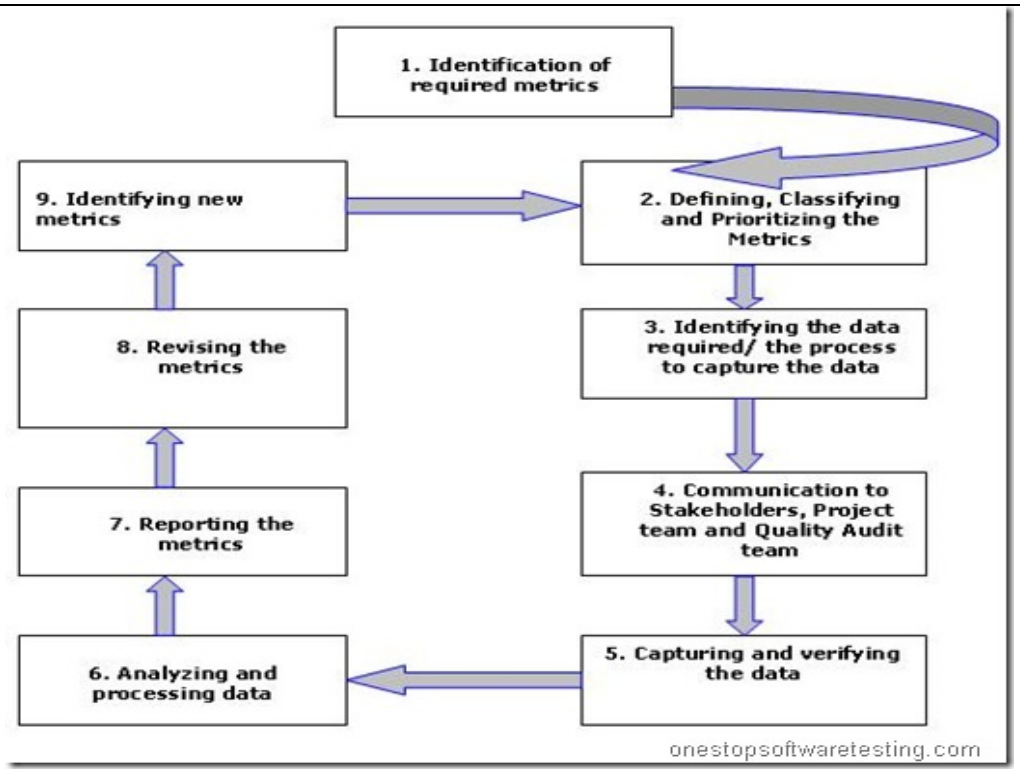
TestComplete is an automated testing tool that lets you create, manage and run tests for any windows, web or rich client software. It makes it easy for anyone to create automated tests. Some features are open APIs, easy extensibility, tons of documentation, scripted testing for total flexibility, windows and web testing, application support etc. It is an easy to use, all-in-one package that lets anyone start automating tests in minutes with no special skills. It has a low price, powerful features and impressive support resources.

6. **TESTDRIVE**



TestDrive is a full-featured automated testing solution designed to test GUI and browser applications "out-of-the-box". Significant reductions in timescales and advanced levels of quality can be achieved without the

		complexity of traditional testing tools. TestDrive integrates with all the other elements of our solution suite so that tests can be run from within Qualify, scripts can be built automatically from manual test results within TestDrive-Assist, and effects in the database can be simultaneously verified in TestBench. **Refer following Link for Testing Using Automated Tools-** • **http://en.wikipedia.org/wiki/Test_automation** • **http://appaloud.com/best-tools-for-test-automation/**
5	**6.6**	**What are Metrics and Measurement** (Ref. book no. 1 Chapter 17 Page 420) An important deliverable from the test team is information that conveys the progress of test work and the status of test activities within the project. This information can include metrics, statistics and summaries and take the form of graphs, charts and written reports. Various partners within the business require this information in support of their own activities and decision making. As such any information provided by the test team should be useful to the business partner and be published in a timely and consistent manner. This discussion document outlines the key Metrics that will be provided by the test team, where the information will be drawn from and how it will be published. The document excludes statistics and summaries except where this is mentioned for context or reference. The term used to describe a measurement of a particular attribute of a software project is a Software Metric. The Software Metrics that the QA team produces are concerned with the test activities that are part of the Test Phase and so are formally known as Software Testing Metrics. Other teams will produce and publish their own Software Metrics during the project. Generation of Software Test Metrics is the most important responsibility of the Software Test Lead/Manager. **Test Metrics are used to,** 1. Take the decision for next phase of activities such as, estimate the cost & schedule of future projects. 2. Understand the kind of improvement required to success the project. 3. Take decision on process or technology to be modified etc.



Importance of Software Testing Metrics:

As explained above, Test Metrics are the most important to measure the quality of the software. Now, **how can we measure the quality of the software by using Metrics**?

Suppose, if a project does not have any metrics, then how the quality of the work done by a Test analyst will be measured?

For Example: A Test Analyst has to,

1. Design the test cases for 5 requirements
2. Execute the designed test cases
3. Log the defects & need to fail the related test cases
4. After the defect is resolved, need to re-test the defect & re-execute the corresponding failed test case.

In above scenario, if metrics are not followed, then the work completed by the test analyst will be subjective i.e. the test report will not have the proper information to know the status of his work/project.

If Metrics are involved in the project, then the exact status of his/her work with proper numbers/data can be published .**i.e. in the Test report, we can publish:**

1. How many test cases have been designed per requirement?
2. How many test cases are yet to design?
3. How many test cases are executed?

4. How many test cases are passed/failed/blocked?
5. How many test cases are not yet executed?
6. How many defects are identified & what is the severity of those defects?
7. How many test cases are failed due to one particular defect? etc.

Based on the project needs we can have more metrics than above mentioned list, to know the status of the project in detail.

Based on the above metrics, test lead/manager will get the understanding of the below mentioned key points.

a) %ge of work completed
b) %ge of work yet to be completed
c) Time to complete the remaining work
d) Whether the project is going as per the schedule or lagging? etc.

Based on the metrics, if the project is not going to complete as per the schedule, then the manager will raise the alarm to the client and other stake holders by providing the reasons for lagging to avoid the last minute surprises.

Refer following Link for Matrix and Measurement-

- **www.rbcs-us.com/images/documents/Metrics-Article2-0711.pdf**
- **www.bth.se/fou/cuppsats.nsf/all/.../$file/Master_Thesis.pdf**

Refer PPT From:

1. **https://www.cs.purdue.edu/homes/apm/slides/uml-slides/metrics.ppt**
2. **www.sdml.info/collard/seF08/notes/Software%20Metrics.ppt**
3. **www.slideshare.net/swatisinghal/software-metrics-5079475**

Types of Metrics

(Ref. book no.1 Chapter 17 Page 427)

Project Metrics:

This category includes status of the project including milestones, budget and schedule variance and project scope changes.

The following are examples of project metrics:

a) Percent of budget utilized
b) Days behind or ahead of schedule
c) Percent of change of project scope
d) Percent of project completed (not a budget or schedule metric, but rather an assessment of the functionality/structure completed at a given point in time)

- **Size Measurements**:

This category includes methods primarily developed for measuring the software size of software systems, such as lines of code, and function points. These can also be used to measure software-testing productivity. Sizing is important in normalizing data for comparison to other projects.

The following are examples of size metrics:

a) KLOC: thousand lines of code, used primarily with statement level languages.
b) Function points: a defined unit of size for software.
c) Pages or words of documentation

- **Product Measures**:

  This category includes measures of a product's attributes such as performance, reliability, usability.

  The following are examples of product measures:

  - **Defect density:** the expected number of defects that will occur in a product during development.
  - **Satisfaction Metrics:** This category includes the assessment of customers of testing on the effectiveness and efficiency of testing.

  The following are examples of satisfaction metrics:

  a) Ease of use: the amount of effort required to use software and/or software documentation.

  b) Customer complaints: some relationship between customer complaints and size of system or number of transactions processed.

  c) Customer subjective assessment: a rating system that asks customers to rate their satisfaction on different project characteristics on a scale, for example a scale of 1-5.

  d) Acceptance criteria met: the number of user defined acceptance criteria met at the time software goes operational.

  e) User participation in software development: an indication of the user desire to produce high quality software on time and within budget.

Project Metrics

(Ref. book no. 1, Chapter 17 Page 428)

- **Effort Variance(Planned vs Actual)-**

  Effort was defined as the total amount of time for a task that results in a work product or a service. The planned amount of time is called '*Planned Effort*' and the actual amount of time spent is called the '*Actual Effort*'; it can be measure in hours or in days depending on the specifics of the project.

  A task is a part of a set of actions which accomplishes a job, problem or assignment which should not be different for Waterfall or Agile projects. However we do acknowledge differences on how tasks are managed on using different methodologies; which can be seen as advantages or disadvantages of each. The following figure shows how this metric is summarized to the team.

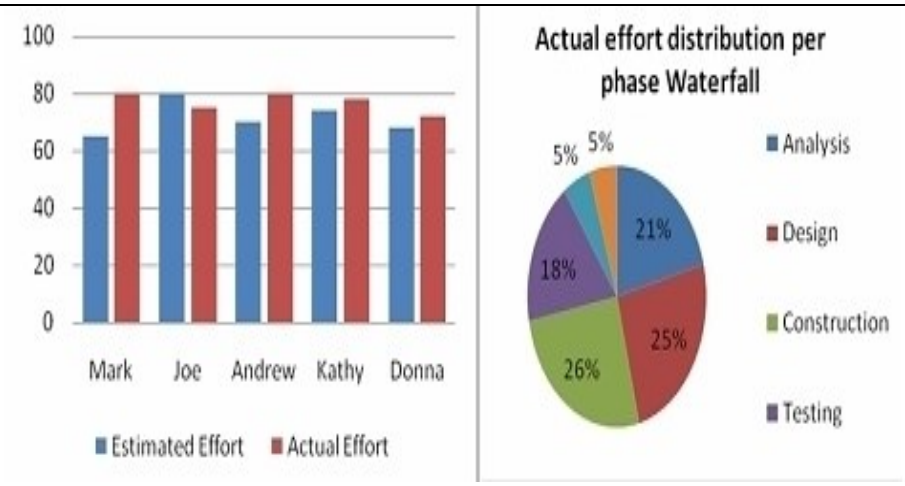


Figure : Cumulative Person Effort Visual Display & *Actual Effort* distribution per phase

Effort is a directly observable quantity which is measured cumulatively for the tasks being assigned to specific resources or it can also be computed for specific tasks, milestones or phases. The '*Planned Effort*' is collected when the work for a specific task is planned and is refined when the task is analyzed or designed. The '*Actual Effort*' is collected during the construction, testing and warranty related to the specific task.

At the beginning organizations should see big differences or gaps between their '*estimation*' and '*actuals*' however as teams "mature" the numbers tend to get closer. Our experience has been that if after a six month period the effort charts don't show closer trends the team must retrospect and find root causes, which can be related to factors inside or outside the team.

Difference between the planned outlined effort and the effort required to actually undertake the task is called Effort variance.

Effort variance =
(Actual Effort – Planned Effort)/ Planned Effort x 100

- **Schedule Variance(Planned vs Actual)-**

Schedule Variance is calculated at the end of every milestone to find out how well the project is doing with respect to the schedule.

Any difference between the scheduled completion of an activity and the actual completion is known as Schedule Variance.

Schedule variance = ((Actual calendar days – Planned calendar days) + Start variance)/ Planned calendar days x 100

- **Effort variance for a phase:**

Variance calculation helps in finding out whether commitments are met on time and whether the estimation method works well. In addition, some indications on product quality can be obtained if the effort distribution across

the various phases are captured and analyzed. The deviation between planned and actual effort for various phases within the project.

Effort variance for a phase = (Actual effort for a phase – planned effort for a phase)/ (planned effort for a phase) x 100

Refer following Link for Project Metrics -

- **http://blog.simplilearn.com/project-management/project-and-process-metrics**

Progress and Productivity Metrics-
(Ref. book no. 1, Chapter 17 Page 433-434,448)

- **Progress Metrics**:

This category includes values associated with numbers or types of defects, usually related to system size, such as "defects/1000 lines of code" or "defects/100 function points," severity of defects, uncorrected defects, etc.

The following are examples of defect metrics:

a) Defects related to size of software.
b) Severity of defects such as very important, important, and unimportant.
c) Priority of defects: the importance of correcting defects.
d) Age of defects: the number of days the defect has been uncovered but not corrected.
e) Defects uncovered in testing
f) Cost to locate a defect

- **Productivity Metrics**:

Productivity metrics combine several measurement and parameters with effort spent on the product. They help in finding out the capability of the team as well as for other purposes, such as,

1. Estimating for the new release.
2. Finding out how well the team is progressing, understanding the reasons for variations in result.
3. Estimating the number of defect that can be found.
4. Estimating release date and quality.
5. Estimating the cost involved in the release.

This category includes the effectiveness of test execution.

		Examples of productivity metrics are: a) Cost of testing in relation to overall project costs: assumes a commonly accepted ratio of the costs of development versus tests. b) Under budget/Ahead of schedule. c) Software defects uncovered after the software is placed into an operational status. d) Amount of testing using automated tools. **Refer following Link for Progress and Productivity Metrics -**
6		➢ **Revision of Topic 6**

5.2 Planning and conduct of Test:

- There will be two tests each of 25 marks.
- The test will be conducted as per the MSBTE schedule.
- The schedule of the test and portion shall be declared at least one week in advance.
- The model answers with the marking scheme shall be displayed on the notice board.
- Teacher shall give the feedback to the students about their performance.

5.3 Details about conduct of Assignments. (Feedback shall be given to students)

(Note:- Every individual student shall perform minimum 3 assignments during the course)

- After completion of 2, 4, 6 chapters one assignment should be given.

- It shall be assessed by subject teacher before giving next Assignment.
- Evaluation of Assignment should be done effectively.
- Sample question paper of Software Testing to be solved by every student

SR. NO	TOPIC NO.	ASSIGNMENT STATEMENT (Question will be given by subject teacher on following topics)
1	1.6, 1.7 2.1.2	Skills of software Tester, V-model Structural testing.
2	3.3 4.3	System Testing Test Process
3	5.3 6.5	Techniques for finding defects Use of automated test tool

5.4 STRATEGIES FOR CONDUCT OF PRACTICAL:

5.4.1 Suggestions for effective conduct of practical and assessment:

- Display the Date wise schedule of the experiment to be performed in the Laboratory.
- Before start of any practical Teachers should explain the specific objective of that particular practical.
- Teacher should divide total students into number of group as given in practical manual.
- Teacher should refer the guidelines given in the lab manual.
- Teacher should make the students aware of instructions given in the lab manual.
- Teacher should ensure that the activities given in the Lab Manual are performed by the student and observations should be tabulated.
- Teacher shall assess the performance of students continuously as per norms prescribed by MSBTE CIAAN norms.
- During assessment teacher is expected to ask questions to the students relevant to the experiment conducted during the lab so that students can prepare while submitting record of the practical. Focus should be given on development of enlisted skills rather than theoretical / codified knowledge.

- Available testing tool shall be installed and checked by subject teacher before explaining concerned practical.

List of experiments (as per Lab. Manual)

Sr. no.	Title of Experiments	No. of Hours
1	To Study Software Testing concepts, types and methods.	2
2	To study any one sample system specification and design the test cases for it. (e.g. Student information system, Library system management system, Hospital management system etc.)	2
3	To design test cases for any application such as railway reservation form	2
4	To write test cases on simple calculator application.	2
5	To design test cases for any login form (Eg: Gmail or Yahoo login form)	2
6	To design test cases for mobile phone system (Eg: check battery is inserted in mobile properly, check SIM is inserted properly, check incoming and outgoing call)	2
7	To design test cases for notepad/WordPad/MS-Word application.	4
8	To design test cases for paint application.	4
9	To design test cases for ATM machine.	4
10	Using any testing tool, atomize and run test cases for notepad / WordPad.	4
11	Using any freeware automation testing tool, atomize and run test cases for Ms-Word application	4
	Total	**32**

6. Mode Of Assessment

6.1 Class Test: -It is proposed that there will be two tests each of 25 marks. The tests will be conducted as per MSBTE schedule

6.1.1 Guideline for setting class test question paper:

The following instruction should be followed strictly by the paper setter (subject teacher):

1. The question paper should be set according to the given **Sample Test Paper format**.
2. Question paper for class tests first & second should be strictly based on the given syllabus.
3. Class test Question paper format:
    - Question 1 will be of 9 marks. The student will have to attempt any three out of four. This question will have each bit of 3 marks.
    - Question 2 and 3 will be of 8 marks each.
      Out of which
      The student has to attempt in question no. 2 any 2 out of 3. These questions will have each bit of 4 marks. And
      The student has to attempt in question no. 3 any 1 out of 2. These questions will have each bit of 8 marks.
4. The teacher should ensure that the curriculum is covered by that time.
5. Duration of class test is one hour.
6. Instructions should be given at the top of the paper.

Class Test-I

Syllabus for class test I:

1. Basics of Software Testing
2. Types of testing
3. Levels of Testing and special tests(3.1 to 3.4)

Class Test-II

Syllabus for class test II:

3. Levels of Testing and special tests(3.5)
4. Test Management
5. Defect management
6. Testing tools and measurements

Note: While setting the question paper

40% questions should be based on Remember level (R).

40% questions should be based on Understanding level (U).

20% questions should be based on Application level (A).

Sample Test Paper

Sample Test Paper (Test 1)

Course Name: **Computer Engineering Group** Course Code: **CO/CM/CD**

Semester: Sixth for CO/CM and Seventh for CD

Subject**: Software Testing** Subject Code: **17624**

Marks: **25** Time: **1 hour**

Instructions:

1. All questions are compulsory
2. Illustrate your answers with neat sketches wherever necessary
3. Figures to the right indicate full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q1. Attempt any THREE of the following: **9**

a) What are objectives of software testing?

b) State Code functional testing with example

c) Mention the situation where load testing is applicable?

d) List any four skills of Software tester.

Q2. Attempt any TWO of the following **8**

a)Write the process of incremental Integration testing.

b) What is test case? Give example of any test case with specifications.

c) State decision table? Draw decision table with its components.

Q3. Attempt any ONE of the following **8**

a) When to start and when to stop testing of software (entry and Exit Criteria)

b) Describe followings with respect to system testing:

i) Load Testing

ii) Stress Testing

iii)Usability Testing

iv)Recovery Testing

Sample Test Paper

Sample Test Paper (Test 2)

Course Name: **Computer Engineering Group** Course Code: **CO/CM/CD**

Semester: Sixth for CO/CM and Seventh for CD

Subject**: Software Testing** Subject Code: **17624**

Marks: **25** Time: **1 hour**

Instructions:

1. All questions are compulsory
2. Illustrate your answers with neat sketches wherever necessary
3. Figures to the right indicate full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q1. Attempt any THREE of the following: 9

a) Mention Classification of product metrics?

b) Which are three techniques for finding defects?

c) Draw labeled diagram of defect management process? List any three characteristics of defect management process

d) List any six tasks contained in unit test plan document.

Q2. Attempt any TWO of the following 8

a)Describe the process of developing test strategy under test planning?

b)List states of Defect life cycle. Draw labeled diagram of defect life cycle.

c)State responsibilities of different types of tester in testing group hierarchy?

Q3. Attempt any ONE of the following 8

a) Which are limitations of manual testing? State any four benefits of automated testing?

b) What are the types of test report? Write contents of Test summary report.

Scheme – G

Sample Question Paper

Course Name: - Computer Engineering Group
Course Code:- CO/CM/CD
Semester: - Sixth for CO/CM and Seventh for CD
Subject Title: - SOFTWARE TESTING **17624**
Marks: - 100 Marks

Instructions

1. All questions are compulsory
2. Illustrate your answer with neat sketches wherever necessary
3. Figures to the right indicates full marks
4. Assume suitable data if necessary
5. Preferably, write the answers in sequential order

Q.1a) Attempt any THREE of the following **12 Marks**

a) Draw labeled diagram and describe Entry Task Verification Exit model for effective verification and validation.
b) Describe formal Inspection under Static testing.
c) Distinguish between Alpha testing and Beta testing with respect to concept, Principle, Environment, participants
d) What is Test Plan? Write any two advantages of test planning.

Q.1b) Attempt any ONE of the following **06 Marks**

a) Draw a sample Admission form for college admission. Prepare six test cases for performing Graphical User Interface Test.
b) Enlist any six attributes of defect. Describe them with example?

Q.2 Attempt any FOUR of the following **16 Marks**

a) State process of Black box testing with labeled diagram? List any four techniques of black box testing.
b) Describe steps in selecting automated testing tool?
c) Explain with diagram three methods of object oriented testing.
d) Which are any four benefits of automation in testing?
e) Draw the diagram of Defect Life Cycle and explain its process.
f) Draw Classification of White Box Testing? Explain any one type of white box testing in detail.

Q.3 Attempt any FOUR of the following **16 Marks**

a) Illustrate process of Equivalence Partitioning with example.
b) State process of Bi-Directional/Sandwich Integration testing with labeled diagram. Give its any two advantages and disadvantages.
c) Which are the factors considered to decide Test approach or Test strategy?

d) Identify any two situations where regression testing can be done? Give importance of regression testing.

e) Define software Metrics? Describe Product vs. Process metrics and Objective vs. Subjective metrics.

Q.4 a) Attempt any THREE of the following **12 Marks**

a) Which information a meaningful test report should contain? Mention purpose of test report.

b) Explain activities performed by Static and Dynamic testing tools.

c) Illustrate defect fixing process with diagram? Give any four examples of critical risk involved in fixing defect.

d) State the factors that need to be considered to identify resource requirement by test manager? List five test deliverables for test plan.

Q.4b) Attempt any ONE of the following **06 Marks**

a) Explain V-model with labeled diagram.

b) State two objectives of user documentation testing? Mention any benefits of user documentation testing?

Q.5 Attempt any TWO of the following **16 Marks**

a) Describe the Test Management Process and give details of following Internal standards for process and method

1. Naming and storage convention
2. Documentation standard

b) Enlist any six parameters that generally affect compatibility of product? Explain Forward compatibility testing.

c) Explain following concepts related to Web-Based testing.

i) Interface Testing

ii) Usability testing

Q.6 Attempt any FOUR of the following **16 Marks**

a) What are the Need of Automation testing?

b) What is significance of Quality Assurance and Quality Control?

c) State how to minimize risk impact while estimating defect?

d) Write test cases for identity card of student having following details:

Name, Surname, Roll no., Date of Birth, Branch, Contact no. , Blood group, Year

e) How to perform security testing? State any four elements of security testing.