SHEME OF VALUATION

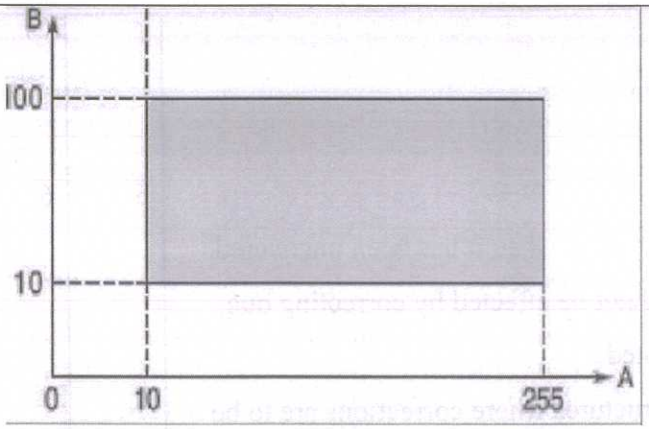
(Scoring Indicators)

Revision: 2015 **Course Code: 6132**

Course Title: SOFTWARE TESTING

Qst. No	Scoring Indicator	Split up score	Sub Total	Total
	PART A			
I(i)	Quality, customer satisfaction, risk management	2	2	
I (ii)	White box testing and black box testing	2	2	
I(iii)	Software was an automated functional GUI testing tool that allowed a user to record and play back user interface (UI) interactions as test scripts.	2	2	10
I(iv)		2	2	
I(v)	Debugging is the process of finding and resolving of defects or problem within in the program Finite state machine	2	2	
	PART B			
II(i)	Verification of code: Check that every design specification in HLD and LLD has been coded using traceability matrix. Examine the code against a language specification checklist. Code verification can be done most efficiently by the developer, as he has prepared the code. He can verify every statement, control structure, loop, and logic such that every possible method of execution is tested. In this way, he verifies the whole module which he has developed. Some points against which the code can be verified are: (a) Misunderstood or incorrect arithmetic precedence (b) Mixed mode operations (c) Incorrect initialization (d) Precision inaccuracy (e) Incorrect symbolic representation of an expression, etc. Two kinds of techniques are used to verify the coding: (a) static testing, and (b) dynamic	3		

	testing. Static testing techniques - this technique does not involve actual execution. It considers only static analysis of the code or some form of conceptual execution of the code. Dynamic testing techniques It is complementary to the static testing technique. It executes the code on some test data. The developer is the key person in this process who can verify the code of his module by using the dynamic testing technique. Unit verification- Verification of coding cannot be done for the whole system. Moreover, the system is divided into modules. Therefore, verification of coding means the verification of code of modules by their developers. Interfaces are verified to ensure that information properly flows in and out of the program unit under test. The local data structure is verified to maintain data integrity. Boundary conditions are checked to verify that the module is working fi ne on boundaries also. All independent paths through the control structure are exercised to ensure that all statements in a module have been executed at least once. All error handling paths are tested. Unit verification is largely white-box oriented. Validation- To determine whether the product satisfies the users' requirements, as stated in the requirement specification. To determine whether the product's actual behavior matches the desired behavior. The bugs that are still present in the software after the coding phase need to be uncovered. Validation testing provides the last chance to discover bugs, otherwise these bugs will move to the final product released to the customer. Validation enhances the quality of software. The validation activities are divided into Validation Test Plan and Validation Test Execution: Validation Test Plan It starts as soon as the first output of SDLC, Include-Acceptance test plan -This plan is prepared in the requirement phase according to the acceptance criteria prepared from the user feedback. System test plan- This plan is prepared to verify the objectives specified in the SRS. function test plan-.			
II(ii)	Integration test plan This plan is prepared to validate the integration of all the modules such that all their interdependencies are checked Maximize the probability of finding errors, test cases are designed with boundary input values. BVA is applicable when the module to be tested is a function of several independent variable. For example if A is an integer between 10 and 255 then boundary checking can be on10(9,10,11) and on 255(256,255,254).	3	6	



4expl

2fig

6

Boundary value analysis

Several methods

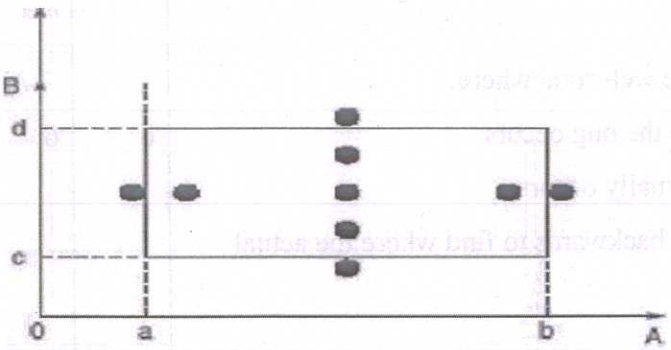
1. Boundary value checking (BVC)- test cases are designed by holding one variable at its extreme value and other variables at normal value.

Minimum value(min)

Value just above the minimum value(min+)

Maximum value(max)

Value just above the minimum value(max-)



2.Robustness testing methods-BVC can be extended such that boundary values are exceeded as

A value just greater than maximum value(max+)

A value just less than minimum value(min-)

3. Worst case testing method- again extended the concept of BVC by assuming more than one variable on the boundary.

4. Robust worst case testing method- the worst case can further be extended if we consider robustness .

II(iii)

The following guidelines to be followed, if planned for automation in testing otherwise may not provide desired solution.

1.Concider building a tool instead of buying one, if possible

2. the tool on an application prototype

	3.not all the test should be automated 4.select the tools according to organizational needs 5.use proven test script development technique 6. Automate the regression test whenever feasible.	6	6	
II(iv)	The second phase of debugging process is to correct error when it has been uncovered. It is not easy process, the design of software should not be effected by correcting bug, Before correcting bug the following points to be noted a. evaluate the coupling of the logic and the data structures where corrections are to be made. b. after recognizing the influence of corrections on other module, plan regression test. C .perform regression testing Steps for correcting bugs Step 1: Enter the bug in your case tracking system You should record these three things in each bug report: 1. What the user was doing 2. What they were expecting 3. What happened instead These will tell you how to recreate the bug. Step 2: Google the error message give you a search query to find the solution on the web somewhere. Step 3: Identify the immediate line of code where the bug occurs Step 4: Identify the line of code where the bug actually occurs Once you know the immediate line, you can step backwards to find where the actual bug occurs. Step 5: Identify the species of bug-for example Configuration or constants are wrong Unexpected null Bad input Wrong precision Coincidences in the development environment didn't carry over to production Assignments instead of comparisons Buffer overflow & Index Out-of-rang Programmer can't do math If your bug doesn't resemble any of the above, or you aren't able to isolate it to a line of code, you'll have more work to do. Step 6: Use the process of elimination Step 7: Log everything and analyze the logs Step 8: Eliminate the hardware or platform as a cause	6	6	30

Step 9: Look at the correlations

Step 10: Bring-in outside help

Your final step will be to reach out to people who know more than you

II(v) These are beneficial for the applications which can be described using state transition diagrams and state tables.

terms related to state tables

An F SM is a behavioral model whose outcome depends upon both previous and current inputs. FSM models can be prepared for software structure or software behavior

State graph or state transition diagrams- A system or its components may have a number of states depending on its input and time. For example, a task in an operating system can have the following states:

1. New State: When a task is newly created

. 2. Ready: When the task is waiting in the ready queue for its turn.

3. Running: When instructions of the task are being executed by CPU.

4. Waiting: When the task is waiting for an I/O event or reception of a signal.

5. Terminated: The task has finished execution. States are represented by nodes. Now with the help of nodes and transition links between the nodes, a state transition diagram or state graph is prepared. A state graph is the pictorial representation of an FSM. Its purpose is to depict the states that a system or its components can assume. It shows the events or circumstances that cause or result from a change from one state to another, it is called a transition. Transitions are represented by links that join the nodes.

6

4+ 2fig

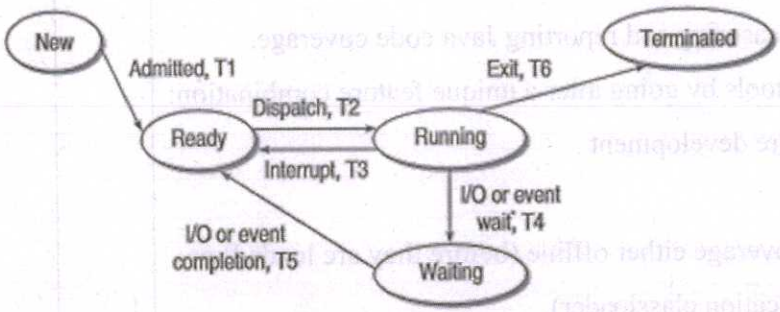


Figure 4.5 State graph

STATE TABLE State graphs of larger systems may not be easy to understand. Therefore, state graphs are converted into tabular form for convenience sake, which are known as state tables.

The procedure for converting state graphs and state tables into test cases is

1. Identify the states
2. Prepare state transition diagram after understanding transitions between states
3. Convert the state graph into the state table as discussed earlier
4. Analyse the state table for its completeness

30

5. Create the corresponding test cases from the state table

Table 4.1 State table

State\Input Event	Admit	Dispatch	Interrupt	I/O or Event Wait	I/O or Event Wait Over	Exit
New	**Ready/ T1**	New / T0	New / T0	New / T0	New / T0	New / T0
Ready	Ready/ T1	**Running/ T2**	Ready / T1	Ready / T1	Ready / T1	Ready / T1
Running	Running/T2	Running/ T2	**Ready / T3**	**Waiting/ T4**	Running/ T2	**Terminated/T6**
Waiting	Waiting/T4	Waiting / T4	Waiting/T4	Waiting / T4	**Ready / T5**	Waiting / T4

II(vi)

Cgreen is a unit tester for the C and C++ software developer, a test automation and software quality assurance tool for programmers and development teams. The tool is completely open source published under the ISC, OpenBSD, license.

Unit testing is a development practice popularised by the agile development community. It is characterised by writing many small tests alongside the normal code. Often the tests are written before the code they are testing, in a tight test-code-refactor loop. Done this way, the practice is known as Test Driven Development.

Here are some of its features:

□ Fluent API resulting in very readable tests

□ Expressive and clear output using the default reporter

□ Fully functional mocks, both strict and loose

2

Emma: written in pure java for measuring and reporting java code coverage

Supported coverage types-class ,method, line , basic block

EMMA is an open-source toolkit for measuring and reporting Java code coverage.

EMMA distinguishes itself from other tools by going after a unique feature combination: support for largescale enterprise software development

EMMA features at a glance:

□ EMMA can instrument classes for coverage either offline (before they are loaded) or on the fly (using an instrumenting application classloader).

□ Supported coverage types: class, method, line, basic block. EMMA can detect when a single source code line is covered only partially.

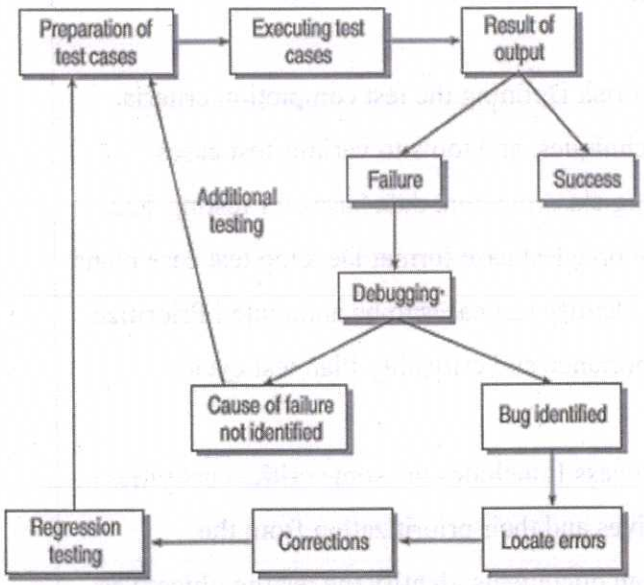
□ Coverage stats are aggregated at method, class, package, and "all classes" levels. □ report types: plain text, HTML, XML.

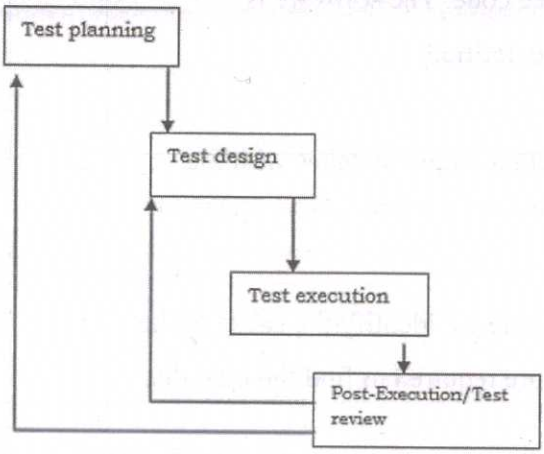
2

6

Findbugs: to inspect java byte code occurrence of bug patterns

FindBugs is an open-source static code analyser created by Bill Pugh and David Hovemeyer which detects possible bugs in Java programs.Potential errors are classified in four ranks: (i) scariest, (ii) scary, (iii) troubling and (iv) of concern. This is a hint to the developer about their possible impact or severity

	FindBugs operates on Java byte code rather than source code. The software is distributed as a stand-alone GUI application bug will be notified.			
II(vii)	Debugging process is explained in the following steps 1. check the output by executing the test cases. if the actual output matches with the expected one ,it means that the result are successful. Otherwise a failure that need to be analyzed 2. debugging is performed for the analysis of failure, where we identify the cause of the problem and correct it.in some cases additional testing are required to find the bug, that is the case cause of failure not identified in fig 3.if symptoms are sufficient to identified the bug then the bug is traced 4. ones the bug is located, then bug is removed with corrections. 5. finally regression testing is performed to check whether the modification effected any other parts of the system.	2 4	6	
	 Figure 18.1 Debugging process	2fig		
	PART C			15
		3		

III	 Software testing life cycle. STLC is simply a testing phase in the SDLC development. Validation and Authentication is tried and tested in this phase. STLC has fixed phases 1. TEST PLANNING – Preparing the test strategy & planning Test planning consists of Defining the test strategy- Estimate the no.of test cases, their duration, cost, Plan the resources Identify areas of risk Defining the test completion criteria Identification of methodologies, techniques, and tools to various test cases Identifying reporting procedures, bug classification, data bases for testing The major outputs of test planning Develop a test case format Develop test case plans according to every phase of SDLC Identify test cases to be automated Prioritize the test cases according to their importance and criticality Plan test cycles required for regression testing 2. TEST Design –it is well planned process It includes the some critical activities such as Determining the test objectives and their prioritization from the requirements specification and design documents identify the testing objectives. Depending on the scope and risk give prioritize the test objectives Preparing list of items to be tested Objects are converted into list of objects Mapping items to test cases Need to create a matrix for knowing which test case is will be covered by which item The matrix will help in i) Identify the major test scenario ii) Reducing the redundant test cases iii) Identifying the absence of a test case for particular objective and as a result, creating them The tester who designs the test cases must understand the cause-andeffect connections also. 3. Test Execution- Understanding the bug- Reproducing the bug- Analyzing the nature and cause of the bug Software testing methodology is the organization of software testing by means of which the test strategy and test tactics are achieved.	5 5 2	15	

IV

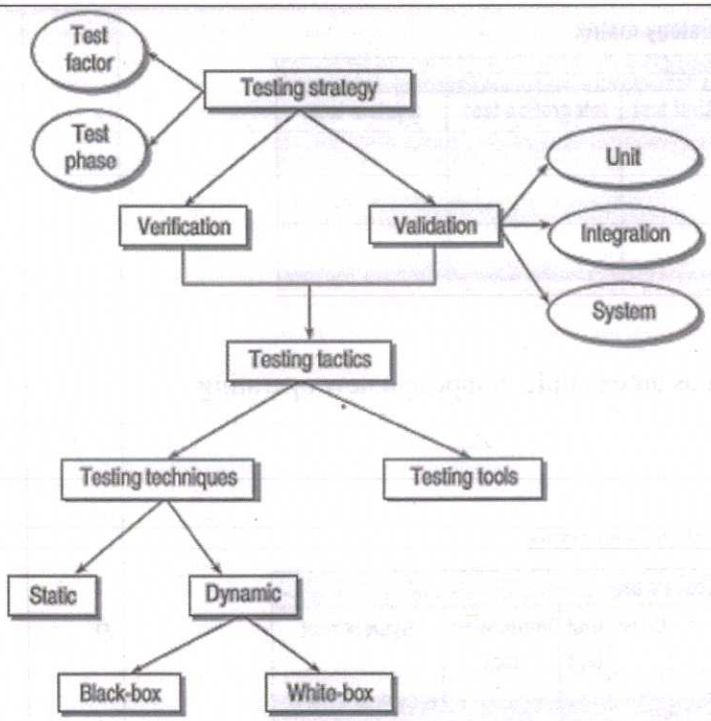


Figure 2.11 Testing methodology

3fig

6

Testing strategy is the planning of the whole testing process into a well-planned series of steps.

Test factors are risk factors or issues related to the system under development. Risk factors need to be selected and ranked according to a specifi c system under development.

Test factors are risk factors or issues related to the system under development. Risk factors need to be selected and ranked according to a specifi c system under development.

15

A test strategy matrix identifi es the concerns that will become the focus of test planning and execution. In this way, this matrix becomes an input to develop the testing strategy. The matrix is prepared using test factors and test phase.

The steps to prepare this matrix are discussed below.

Select and rank test factors Based on the test factors list. These are the rows of the matrix.

Identify system development phases Different phases according to the adopted development model are listed as columns of the matrix. These are called test phases.

Identify risks associated with the system under development In the horizontal column under each of the test phases, the test concern with the strategy used to address this concern is entered.

Table 2.2 Test strategy matrix

Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test

Creating a test strategy Let's take a project as an example. Suppose a new operating system has to be designed.

Table 2.3 Example test strategy matrix

Test Factors	Test Phase					
	Requirements	Design	Code	Unit test	Integration test	System test
Portability	Is portability feature mentioned in specifications according to different hardware?					Is system testing performed on MIPS and INTEL platforms?
Service Level	Is time frame for booting mentioned?	Is time frame incorporated in design of the module?				

Select and rank test factors A critical factor to be considered for the development of an operating system is portability. This is the effort required to transfer

Identify the test phases In this step, all test phases affected by the selected test factors are identified.

Identify the risks associated with each test factor and its corresponding test phase.

Plan the test strategy for every risk identified.

DEVELOPMENT OF TEST STRATEGY- Verification and Validation—the basis for any type of testing.

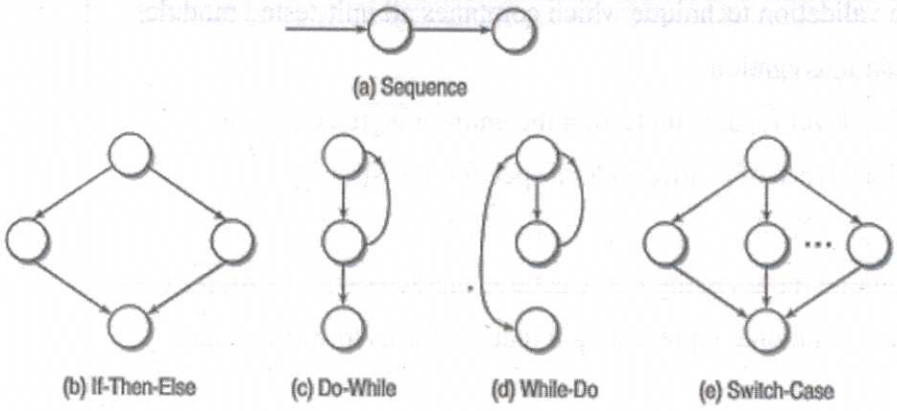
Check every sub-task to ensure that you are working in the right direction. This is verification. After the sub-tasks have been completed and merged, the entire task is checked to ensure the required task goals have been achieved. This is validation.

Verification and validation (V&V) are the building blocks of a testing process. The formation of test strategy is based on these two terms only. V&V can be best understood when these are modeled in the testing process. This model is known as the Testing Life Cycle Model.

Validation has the following three activities which are also known as the three levels of validation testing.

Unit Testing It is a major validation effort performed on the smallest module of the system.

	Integration Testing It is a validation technique which combines all unit-tested modules and performs a test on their aggregation. System Testing This testing level focuses on testing the entire integrated system The ways to perform various types of testing under a specifi c test strategy. The technique used to design effective test case is called Software Testing Technique- Static Testing It is a technique for assessing the structural characteristics of source code, design specifications or any notational representation that conforms to welldefi ned syntactic rules. Dynamic Testing All the methods that execute the code to test a software are known as dynamic testing techniques. Black-box testing This technique takes care of the inputs given to a system and the output is received after processing in the system. White-box testing This technique complements black-box testing. Here, the system is not a black box.	5		
V(a)	Testing Tools Testing tools provide the option to automate the selected testing technique with the help of tools. Basis path testing is the oldest structural testing technique. The technique is based on the control structure of the program. Based on the control structure, a flow graph is prepared and all the possible paths can be covered and executed during testing. CONTROL FLOW GRAPH The control flow graph is a graphical representation of control structure of a program. F low graphs can be prepared as a directed graph. A directed graph (V, E) consists of a set of vertices V and a set of edges E that are ordered pairs of elements of V. Based on the concepts of directed graph, following notations are used for a fl ow graph: Node It represents one or more procedural statements. The nodes are denoted by a circle. These are numbered or labeled. Edges or links They represent the flow of control in a program. This is denoted by an arrow on the edge. An edge must terminate at a node. Decision node A node with more than one arrow leaving it is called a decision node Junction node A node with more than one arrow entering it is called a junction. Regions Areas bounded by edges and nodes are called regions, FLOW GRAPH NOTATIONS FOR DIFFERENT PROGRAMMING CONSTRUCTS the fl ow graph is also known as decision-to-decision-graph or DD graph	3	8	



15

Path A path through a program is a sequence of instructions or statements that starts at an entry, junction, or decision and ends at another, or possibly the same, junction, decision, or exit.

Segment Paths consist of segments

Path segment A path segment is a succession of consecutive links that belongs to some path.

Length of a path.

Cyclomatic complexity- measure for the logical complexity of a program by considering its control fl ow graph

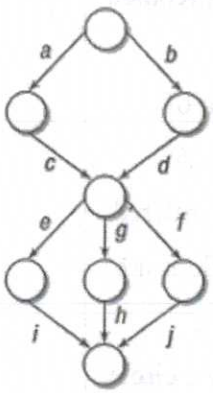


Figure 5.3 Sample graph

if they contain at least one cycle, the number of paths is infi nite. Therefore, he considers only independent paths.

there are six possible paths: acei, acgh, acfh, bdei, bdgh, bdfj.

, the number of independent paths (as a function of the original graph) is given by V(G) = e – n + 2 This is called the cyclomatic number of a program.

Guidelines for Basis Path Testing -We can use the cyclomatic complexity number in basis path testing. 7

Draw the flow graph using the code provided for which we have to write test cases.

Determine the cyclomatic complexity of the fl ow graph. Cyclomatic complexity provides the number of independent paths.

V(b)	1. test cases for black box can be designed earlier than white-box test cases, they cannot be executed until the code is produced and checked using white-box testing techniques. Thus, white-box testing is not an alternative but an essential stage. 2. Since white-box testing is complementary to black-box testing, there are categories of bugs which can be revealed by white-box testing. 3. Errors which have come from the design phase will also be reflected in the code, therefore we must execute white-box test cases for verification of code (unit verification). 4. We often believe that a logical path is not likely to be executed when, in fact, it may be executed on a regular basis. White-box testing explores these paths too. 5. Some typographical errors are not observed and go undetected and are not covered by black-box testing techniques	6	15	
VI	Static testing can be categorized into the following types: Software inspections , Walkthroughs, Technical reviews. An inspection process involves the interaction of the following elements: Inspection steps Role for participants Item being inspected The entry and exit criteria are used to determine whether an item is ready to be inspected. Entry criteria mean that the item to be inspected is mature enough to be used. For the inspection process, a minimum of the following four team members are required. Author/Owner/Producer A programmer or designer responsible for producing the program or document. Inspector A peer member of the team, i.e. he is not a manager or supervisor. He is not directly related to the product under inspection and may be concerned with some other product. He fi nds errors, omissions, and inconsistencies in programs and documents. Moderator A team member who manages the whole inspection process Recorder One who records all the results of the inspection meeting. Advantages- bug reduction, productivity, bug prevention, real time feedback, quality improvement, etc. Structured Walkthrough Though structured walkthrough is a variant of the inspection process. A typical structured walkthrough team consists of the following members: Coordinator Organizes, moderates, and follows up the walkthrough activities. Presenter/Developer Introduces the item to be inspected. This member is optional. Scribe/Recorder Notes down the defects found and suggestion proposed by the members. Reviewer/Tester Finds the defects in the item. Maintenance Oracle Focuses on long-term implications and future maintenance of the project. Standards Bearer Assesses adherence to	6		

standards. User Representative/Accreditation Agent Refl ects the needs and concerns of the user.

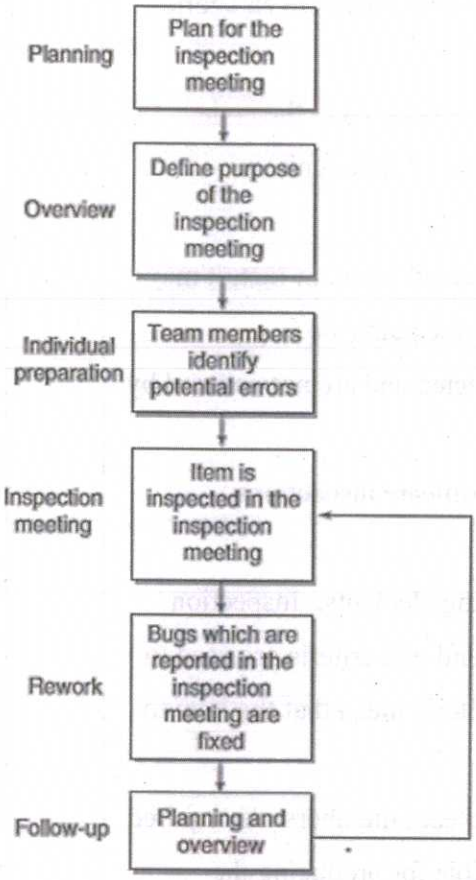


Figure 6.1 Inspection process

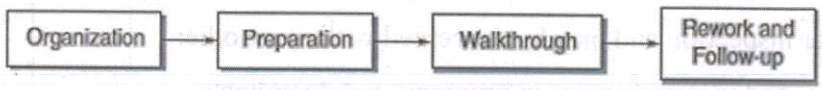


Figure 6.7 Walkthrough process

A technical review is intended to evaluate the software in the light of development standards, guidelines, and specifications and to provide the management with evidence that the development process is being carried out according to the stated objectives. A review is similar to an inspection or walkthrough, except that the review team also includes management. 3

it is considered a higher-level technique as compared to inspection or walkthrough. A technical review team is generally comprised of management-level representatives and project management. Review agendas should focus less on technical issues and more on oversight than an inspection. The purpose is to evaluate the system relative to specifi cations and standards, recording defects and deficiencies. The moderator should gather

	and distribute the documentation to all team members for examination before the review. He should also prepare a set of indicators to measure the following points: Appropriateness of the problem definition and requirements Adequacy of all underlying assumptions. Adherence to standards, Consistency, Completeness, Documentation. The moderator may also prepare a checklist to help the team focus on the key points.			
VII	The different categories of testing tools are static and dynamic testing tools- static testing tool-This is the opposite of dynamic testing whereby it checks the systems software and does not execute the program itself. Static testing tool scans the program and detect possible faults and anomalies, and detect the following 1.statements are well formed 2.interferances about the control flow of the program 3.compute the set of all possible values And performs the following types of static analysis. Control flow analysis Data use analysis Interface analysis Path analysis Dynamic testing tool-support the following testing activities Dynamic testing activities Enable the test team to capture events during execution List the number of times a line of code is executed Testing activity tools-testing activities are categorised in to four and the corresponding tools are 1..reviews and inspections tools-this tools are for static analysis following types of tools are required a.complexity analysis tools-complexity can be analysed so that testing time and resources can be estimated b.code comprehension-helps in understanding dependencies 2.tools for test planning-the required tools are tools for test plan documentation,tools for test schedule, tools for complexity analyser .tools for test design and development-the types of tools required are a.test data generator-it automates the generation of test data baseed on user defined format.	6 6	15	

	b.test case generator-it automates the procedure for generating test cases. 3.test execution and evaluation –types of tools required for test execution and evaluation are a.playback tools-this tools record events(including keystroke,mouseclick)at the time of running and record the information,the tool replay test by using this information. b.coverage analysis tool-provide a quatitative measure of the coverage of the system being tested. Measuring structural coverage Measuring the number of integration Quantifying complexity Measuring multiples levels of test coverage Memory testing tool-this tools verify that an application is properly using its memory resources. Network testing tool-this tools monitor, measure, test, and diagnose performance across entire network. Unit Testing Tools This is used in the testing process of the method. JUnit is a very good testing tool for using with unit testing. It is a java-based tool, which is capable of going through code especially with extreme programming as it takes up less time to do. Regression Testing Tools Many different tools can be used for regression. It helps to automate the testing as the program can be reused. An example of a program is configuration testing which is a program that is run with a new device or a new version of the operating system with a new application. It's a bit like port testing. Web Tools This is used over web based applications. Security Testing Tools These can help aid with the security of the system. It can help against unauthorized access, hacking and any coding damage which deals with the code of application. It uses very sophisticated testing techniques. QA inspect is a very good program to use for this type of testing. Functional Testing Tools These are tools which help with the system testing. An example of a program is Badboy which provides an enhanced browser interface to help aid in building and testing dynamic applications. It allows the developer to monitor and understand the interactions between the browser and the server.	3		

	Performance Testing Tools This tool is used for web servers and to analyze the performance and characteristics of the web application. It is similar to using web tools but has its own applications. An example program is LoadTracer which is a GUI tool and checks the loading and performance and scalability of the web application. Database Testing Tools Database testing tools are very good to use for checking and testing databases. Web based testing- Used for testing web based systems ,have different nature from traditional .due to environment difference and challenges of dynamic behavior , complexity and diversity makes testing a challenge. Web application testing, a software testing technique exclusively adopted to test the applications that are hosted on web in which the application interfaces and other functionalities are tested.			
VIII(a)	Web Application Testing - Techniques: 1. Functionality Testing - The below are some of the checks that are performed but not limited to the below list: □ Verify there is no dead page or invalid redirects. □ First check all the validations on each field. □ Wrong inputs to perform negative testing. □ Verify the workflow of the system. □ Verify the data integrity. 2. Usability testing - To verify how the application is easy to use with. □ Test the navigation and controls. □ Content checking 3. Interface testing - Performed to verify the interface and the dataflow from one system to other. 4. Compatibility testing- Compatibility testing is performed based on the context of the application. □ Browser compatibility □ Operating system compatibility □ Compatible to various devices like notebook, mobile, etc. 5. Performance testing - Performed to verify the server response time and throughput under various load conditions. Load testing - It is the simplest form of testing conducted to understand the behaviour of the system under a specific load. Load testing will result in measuring important business critical transactions and load on the database, application server, etc. are also monitored. Stress testing - It is performed to find the upper limit capacity of the system and also to determine how the system performs if the current load goes well above the expected	8		

	maximum. Soak testing - Soak Testing also known as endurance testing, is performed to determine the system parameters under continuous expected load. During soak tests the parameters such as memory utilization is monitored to detect memory leaks or other performance issues. The main aim is to discover the system's performance under sustained use. □ Spike testing - Spike testing 6. Security testing - Performed to verify if the application is secured on web as data theft and unauthorized access are more common issues and below are some of the techniques to verify the security level of the system. 1. Interface testing: user interface with web application must be proper and flexible. The interface between the concerned client and server also considered .two main interface web server and application server interface, application server and database server interface. 2. Usability testing: web application is reviewed and tested from a user's viewpoint. We can even lose users because of poor design. 3. Content testing: to check the completeness, correctness properties of web application content, if these content satisfactory to users they may not visit the web page again. 4.navigation testing: to ensure the functioning of correct sequence of those navigations, the following navigations are correctly executing internal links, external links, redirected links ,etc 5. configuration/compatibility testing: there is compatibility between various available resources and application software.			
VIII(b)	Object oriented testing-The shift from traditional to object-oriented environment involves looking at and reconsidering old strategies and methods for testing the software. The traditional programming consists of procedures operating on data, while the object-oriented paradigm focuses on objects that are instances of classes. With the adoption of OO paradigm, almost all the phases of software development have changed in their approach, environments, and tools. Though OO paradigm helps make the designing and development of software easier. OO program should be tested at different levels to uncover all the errors. At the algorithmic level, each module (or method) of every class in the program should be tested in isolation. State-based testing is used to verify whether the methods (a procedure that is executed by an object) of a class are interacting properly with each other. State-based testing is used to verify whether the methods (a procedure that is executed by an object) of a class are interacting properly with each other.	7	15	

	Scenario-based testing is used to detect errors that are caused due to incorrect specifications and improper interactions among various segments of the software. Incorrect interactions often lead to incorrect outputs.			
IX	Debugging technique: 1. debugging with memory dump: take the printout of all registers and relevant memory Locations is obtained and studied. the relevant data of the program is observed through these memory locations and registers for any bug in the program. Not an efficient method, the following are the drawbacks a. it is limited to static state of the program.(state of the program at one instant) b.difficulty in establishing the correspondence between storage locations and variables. 2.debugging with watch points: at a particular point in the program the value of variable or some other action can be verified. 1. Output statement: output statements can be used to check the state of condition or variable at some watch points. output statements are inserted at various watch points the program is compiled and executed with this watch points and output is analyzed it may give some clues. Drawbacks are a. require many changes in the code chance to mask an error or introduce new errors in the program b. after analysing the bug we may forget to remove the output statement 2. Breakpoint execution: break point is actually watch point inserted at various places in the program, Not placed in actual user program therefore no need to remove the breakpoints Advantages over output statement a.no need to compile the program after inserting breakpoints removing breakpoints is easy compared to output statements full program is executed in the case of output statement.but in break point we can control the execution types: unconditional-without any condition we can set brek point conditional-one expression is evaluated if it true brek point cause a stop otherwise execution will continne temporary-used only ones in the program,ones it stops the execution it is removed automatically. internal breakpoints-invisible to the user ,breakpoints set by the debugger itself for	15	15	

	itsown purpose. 3. Single stepping: the users should be able to watch the program after every executable instruction. 4.step into-it means execution proceeds to any function in the source statement and stops at first executable statement in the function. 5.step over-it is also called skip, to skip a portion. 6. Backtracking: Logical approach for debugging, The person debugging must have knowledge about the design of the system. Require manual observation of the code Is very effective when compared to others.			
X	Debuggers: Debugger is a tool to help track down isolate, remove bugs from the software.it allows the programmer to fillow the program execution and at any desired point, stop the program and inspect the state of the program to verify correctness. Is used for a.to illustrate the dynamic nature of a program b.understand a program as well as to find and fix bug c.control over the program under test. Types of debuggers: a.Kernel debugger-A kernel debugger is a debugger present in some operating system kernels to ease debugging and kernel development by the kernel developers. A kernel debugger might be a stub implementing low-level operations The Windows NT family includes a kernel debugger named KD □ Linux kernel; No kernel debugger was included in the mainline Linux b.source-level debugger -A debugger that shows the programmer the line or expression in the source code that resulted in a particular machine code instruction of a running program loaded in memory c. basic machine level debugger-is used for debugging actual running code . d.in circuit emulator-it emulates the system services so that all interactions between an application and the system can be monitored and traced. e.interpretive programming environment debugger- is well integrated in to runtime interpreter and has very tight control over the running application. f. A memory debugger also known as a runtime debugger[1] is a debugger for finding software memory problems such as memory leaks and buffer overflow Debugging tools-The advanced debugger adb is the standard UNIX debugger found on Solaris 1 and 2, HP-UX and SCO. It is the successor of a debugger called db. b. DBG is an open-source debugger and profiler for the PHP programming language. Features □ Remote and local debugging □ Explicit and implicit activation	8		

	□ Call stack, including function calls. c. The GNU Debugger (GDB) is a portable debugger that runs on many Unix-like systems and works for many programming languages, including Ada, C, C++, Objective-C, Free Pascal, Fortran, Go[1] and partially others.[2] Features GDB offers extensive facilities for tracing and altering the execution of computer programs. The user can monitor and modify the values of programs' internal variables, and even call functions independently. Remote debugging GDB offers a "remote" mode often used when debugging embedded systems. d. The modular debugger (mdb) is an extensible, low-level debugger developed by Sun Microsystems for the Solaris 7 operating system. It is now open sourced, under the Common Development and Distribution License (CDDL). Its source code is now available in all open source derivatives of Solaris, such as Illumos. e. The Microsoft Visual Studio Debugger is a debugger that ships along with all versions of Microsoft Visual Studio.	7	15	