MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 01

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I -Testing Techniques & Test Case Design Date of Lecture: 20.08.2021

Topic of Lecture: Using White Box Approach to Test design, Test Adequacy Criteria , Static Testing Vs. Structural Testing

Introduction: One of the basic goals of white box testing is to verify a working flow for an application. It involves testing a series of predefined inputs against expected or desired outputs so that when a specific input does not result in the expected output, you have encountered a bug.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Error
- Detailed requirement
- Functional specifications
- High-level design documents
- Detailed design documents
- Source code

Detailed content of the Lecture:

Using White Box Approach to Test design

- White Box Testing is defined as the testing of a software solution's internal structure, design, and coding.
- In this type of testing, the code is visible to the tester.
- It focuses primarily on verifying the flow of inputs and outputs through the application, improving design and usability, strengthening security.
- White box testing is also known as Clear Box testing, Open Box testing, Structural testing, Transparent Box testing, Code-Based testing, and Glass Box testing. It is usually performed by developers.
- A complementary approach to test case design will be examined where the tester has knowledge of the internal logic structure of the software under test.
- The tester‘s goal is to determine if all the logical and data elements in the software unit are functioning properly. This is called the white box, or glass box, approach to test case design.
- The knowledge needed for the white box test design approach often becomes available to the tester in the later phases of the software life cycle, specifically during the detailed design phase of development.

- Another point of contrast between the two approaches is that the black box test design strategy can be used for both small and large software components, whereas white box-based test design is most useful when testing small components.
- This is because the level of detail required for test design is very high, and the granularity of the items testers must consider when developing the test data is very small.

Test Adequacy Criteria

- The goal for white box testing is to ensure that the internal components of a program are working properly. A common focus is on structural elements such as statements and branches.
- The tester develops test cases that exercise these structural elements to determine if defects exist in the program structure. The term exercise is used in this context to indicate that the target structural elements are executed when the test cases are run.
- By exercising all of the selected structural elements the tester hopes to improve the chances for detecting defects.
- Testers need a framework for deciding which structural elements to select as the focus of testing, for choosing the appropriate test data, and for deciding when the testing efforts are adequate enough to terminate the process with confidence that the software is working properly.

Static Testing Vs. Structural Testing

- Static testing is about prevention whereas dynamic testing is about cure. Static testing is more cost-effective than dynamic testing. Static testing tools provide greater marginal benefits as compare to dynamic testing.
- Static testing gives comprehensive diagnostics for code than dynamic testing. Dynamic testing finds fewer bugs as compare to static testing.
- Dynamic testing usually takes longer time as compare to static testing as it test each case separately.
- Static testing covers more areas than dynamic testing in shorter time. Static testing is done before the code deployment whereas dynamic testing is after the code deployment.
- Static testing is done in verification stage whereas dynamic testing is done in validation stage. In static testing code is being examined without being executed whereas in dynamic testing, code is being executed and tested without necessarily being examined.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Using-white-box-approach-to-test-design_9153/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:47-55)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 02

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 21.08.2021

Topic of Lecture: Code Functional Testing, Coverage and Control Flow Graphs, Covering Code Logic

Introduction: The control flow of programs can be represented by directed graphs. It provides a uniform and detailed formal basis for control flow graphs combining known definitions and results with new aspects. Two graph reductions are defined using only syntactical information about the graphs, but no semantical information about the represented programs. Based on graphs, we define statement coverage and branch coverage such that coverage notions correspond to node coverage, and edge coverage, respectively.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Control Flow Graph
- Functional specifications
- Source code
- Coverage Logics

Detailed content of the Lecture:

Code Functional Testing

- Code Functional Testing is a type of software testing that validates the software system against the functional requirements/specifications. The purpose of Functional tests is to test each function of the software application, by providing appropriate input, verifying the output against the Functional requirements.
- Functional testing mainly involves black box testing and it is not concerned about the source code of the application. This testing checks User Interface, APIs, Database, Security, Client/Server communication and other functionality of the Application Under Test. The testing can be done either manually or using automation.

Coverage and Control Flow Graphs

- The application of coverage analysis is typically associated with the use of control and data flow models to represent program structural elements and data. The logic elements most commonly considered for coverage are based on the flow of control in a unit of code.
- For example,

(i) program statements;

(ii) decisions/branches (these influence the program flow of control);

(iii)conditions (expressions that evaluate to true/false, and do not contain any other true/false-valued expressions);

(iv)combinations of decisions and conditions;

(v) Paths (node sequences in flow graphs).

These logical elements are rooted in the concept of a program prime. A program prime is an atomic programming unit. All structured programs can be built from three basic primes-sequential (e.g., assignment statements), decision (e.g., if/then/else statements), and iterative (e.g., while, for loops). Graphical representations for these three primes.

- Using the concept of a prime and the ability to use combinations of primes to develop structured code, a (control) flow diagram for the soft- ware unit under test can be developed.
- The flow graph can be used by the tester to evaluate the code with respect to its testability, as well as to develop white box test cases.
- For simplicity, sequential statements are often omitted or combined as a block that indicates that if the first statement in the block is executed, so are all the following statements in the block. Edges in the graph represent transfer of control.
- The direction of the transfer depends on the outcome of the condition in the predicate (true or false).

Covering Code Logic

- Logic-based white box-based test design and use of test data adequacy/ coverage concepts provide two major payoffs for the tester:

  (i) Quantitative coverage goals can be proposed, and

  (ii) Commercial tool support is readily available to facilitate the tester‘s work.

- Testers can use these concepts and tools to decide on the target logic elements (properties or features of the code) and the degree of coverage that makes sense in terms of the type of software, its mission or safety criticalness, and time and resources available.
- In terms of a flow graph model of the code, satisfying this criterion requires that all the nodes in the graph are exercised at least once by the test cases if the tests achieve this goal, the test data would satisfy the statement adequacy criterion.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Coverage-and-Control-Flow-Graphs_9155/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, “Software Testing – Principles and Practices”, Pearson Education, 2009 **(Page No:56-64)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 03

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I- Testing Techniques & Test Case Design Date of Lecture: 24.08.2021

Topic of Lecture: Paths Their Role in White box Based Test Design, Code Complexity Testing

Introduction: Metrics can be actionable, but not empirically useful at the same time. McCabe's cyclomatic complexity is one such metric. Broadly speaking, cyclomatic complexity is derived by counting the number of potential paths through the system (typically at the method level). Originally designed to estimate the number of unit tests a method needs, cyclomatic complexity is built into a lot of metric tools and static analysis tools. Developers can and often do use it to measure their notion of "cognitive" complexity, and use it to target their refactoring efforts. Lower the cyclomatic complexity, and improve the testability of your code.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Control Flow Graph
- Functional specifications
- Source code
- Coverage Logics

Detailed content of the Lecture:

Paths Their Role in White box Based Test Design

- The role of a control flow graph as an aid to white box test design was described. It was also mentioned that tools were available to generate control flow graphs.
- These tools typically calculate a value for a software attribute called McCabe's Cyclomatic Complexity V (G) from a flow graph.
- The cyclomatic complexity attribute is very useful to a tester. The complexity value is usually calculated from the control flow graph (G) by the formula

  V(G)=E-N+2

- The value E is the number of edges in the control flow graph and N is the number of nodes. This formula can be applied to flow graphs where there are no disconnected components. As an example, the cyclomatic complexity of the flow graph is calculated as follows:

E=7,N=6

V(G)=7-6+2=3

- The cyclomatic complexity value of a module is useful to the tester in several ways. One of its uses is to provide an approximation of the number of test cases needed for branch coverage in a module of structured code.
- If the testability of a piece of software is defined in terms of the number of test cases required to adequately test it, then McCabe's' cyclomatic complexity provides an approximation of the testability of a module. The tester can use the value of V(G) along with past project data to approximate the testing time and resources required to test a software module.
- In addition, the cyclomatic complexity value and the control flow graph give the tester another tool for developing white box test cases using the concept of a path. A definition for this term is given below.

A path is a sequence of control flow nodes usually beginning from the entry node of a graph through to the exit node.

- An independent path is a special kind of path in the flow graph. Deriving a set of independent paths using a flow graph can support a tester in identifying the control flow features in the code and in setting coverage goals.
- A tester identifies a set of independent paths for the software unit by starting out with one simple path in the flow graph and iteratively adding new paths to the set by adding new edges at each iteration until there are no more new edges to add.
- The independent paths are defined as any new path through the graph that introduces a new edge that has not be traversed before the path is defined.

Code Complexity Testing

- Cyclomatic Complexity in Software Testing is a testing metric used for measuring the complexity of a software program. It is a quantitative measure of independent paths in the source code of a software program.
- Cyclomatic complexity can be calculated by using control flow graphs or with respect to functions, modules, methods or classes within a software program. Independent path is defined as a path that has at least one edge which has not been traversed before in any other paths
- Control flow depicts a program as a graph which consists of Nodes and Edges.In the graph, Nodes represent processing tasks while edges represent control flow between the nodes.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Paths--Their-Role-in-White-Box-Based-Test-Design_9157/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 (Page No:65-68)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 04

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 25.08.2021

Topic of Lecture: Evaluating Test Adequacy Criteria, Test Case Design Strategies

Introduction: A test case includes not only input data but also any relevant execution conditions and procedures, and a way of determining whether the program has passed or failed the test on a particular execution. The term "input" is used in a very broad sense, which may include all kinds of stimuli that contribute to determining program behavior. For example, an interrupt is as much an input as is a file. The pass/fail criterion might be given in the form of expected output, but could also be some other way of determining whether a particular program execution is correct.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Test Case
- Test Suite
- Functional Design
- Control Flow Graph
- Test Harness

Detailed content of the Lecture:

Evaluating Test Adequacy Criteria

- Most of the white box testing approaches we have discussed so far are associated with application of an adequacy criterion.
- Testers are often faced with the decision of which criterion to apply to a given item under test given the nature of the item and the constraints of the test environment (time, costs, resources)
- The criteria at the top of the hierarchy are said to subsume those at the lower levels. For example, achieving all definition-use path adequacy means the tester has also achieved both branch and statement adequacy. Note from the hierarchy that statement adequacy is the weakest of the test adequacy criteria.
- Unfortunately, in many organizations achieving a high level of statement coverage is not even included as a minimal testing goal.

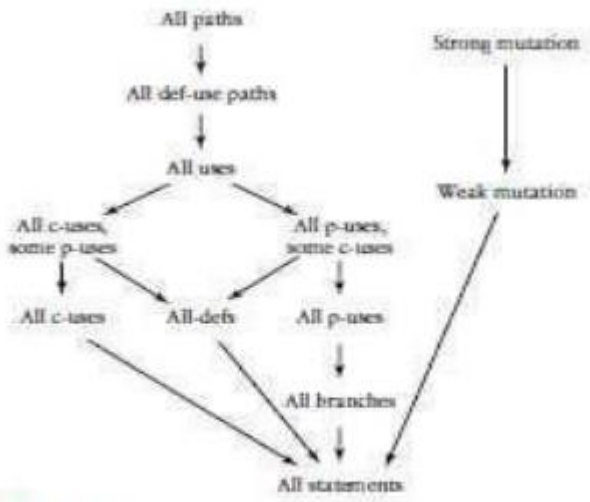


FIG. 5.5
A partial ordering for test adequacy criteria.

- As a conscientious tester you might at first reason that your testing goal should be to develop tests that can satisfy the most stringent criterion. However, you should consider that each adequacy criterion has both strengths and weaknesses.
- Each, is effective in revealing certain types of defects. Application of the so-called stronger criteria usually requires more tester time and resources.
- This translates into higher testing costs. Testing conditions, and the nature of the software should guide your choice of a criterion.
- Support for evaluating test adequacy criteria comes from a theoretical treatment developed by Weyuker.

Test Case Design Strategies

- A smart tester who wants to maximize use of time and resources knows that she needs to develop what we will call effective test cases for execution-based testing. By an effective test case we mean one that has a good possibility of revealing a defect.
- The ability to develop effective test cases is important to an organization evolving toward a higher-quality testing process.
- It has many positive consequences. For example, if test cases are effective there is (i) a greater probability of detecting defects, (ii) a more efficient use of organizational resources, (iii) a higher probability for test reuse, (iv) closer adherence to testing and project schedules and budgets, and, (v) the possibility for delivery of a higher -quality software product.
- What are the approaches a tester should use to design effective test cases? To answer the question we must adopt the view that software is an engineered product. Given this view there are two basic strategies that can be used to design test cases. These are called the black box (sometimes called functional or specification) and white box (sometimes called clear or glass box) test strategies.
- Using the black box approach, a tester considers the software-under test to be an opaque box. There is no knowledge of its inner structure (i.e., how it works).
- The tester only has knowledge of what it does. The size of the software-under-test using this approach can vary from a simple module, member function, or object cluster to a subsystem or a complete Software system.

- The description of behavior or functionality for the software-under-test may come from a formal specification, an Input/Process/Output Diagram (IPO), or a well-defined set of pre and post conditions.
- The tester provides the specified inputs to the software-under-test, runs the test and then determines if the outputs produced are equivalent to those in the specification.
- Because the black box approach only considers software behavior and functionality, it is often called functional or specification-based testing.
- This approach is especially useful for revealing requirements and specification defects.
- The white box approach focuses on the inner structure of the software to be tested. To design test cases using this strategy the tester must have knowledge of that structure.
- The code, or a suitable pseudo code like representation must be available.
- The tester selects test cases to exercise specific internal structural elements to determine if they are working properly.
- Since designing, executing, and analyzing the results of white box testing is very time consuming, this strategy is usually applied to smaller-sized pieces of software such as a module or member function.
- White box testing methods are especially useful for revealing design and code-based control, logic and sequence defects, initialization defects, and data flow defects.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Evaluating-Test-Adequacy-Criteria_9159/

Important Books/Journals for further learning including the page nos.:

Adithya P. Mathur, " Foundations of Software Testing – Fundamentals algorithms and techniques", Dorling Kindersley (India) Pvt. Ltd., Pearson Education, 2008(Page No:415-420)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 05

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 27.08.2021

Topic of Lecture: Using Black Box Approach to Test Case Design, Random Testing, Requirements based testing

Introduction: Black-box testing is a method of software testing that examines the functionality of an application without peering into its internal structures or workings. This method of test can be applied virtually to every level of software testing: unit, integration, system and acceptance.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Requirement Specification
- Test Case
- Security

Detailed content of the Lecture:

Using Black Box Approach to Test Case Design

Black Box Testing, also known as Behavioral Testing, is a software testing method in which the internal structure/design/implementation of the item being tested is not known to the tester.

This method attempts to find errors in the following categories

- Incorrect or missing functions
- Interface errors
- Errors in data structures or external database access
- Behavior or performance errors
- Initialization and termination errors

Random Testing

- Random Testing, also known as monkey testing, is a form of functional black box testing that is performed when there is not enough time to write and execute the tests

Random Testing Characteristics:

- Random testing is performed where the defects are not identified in regular intervals.
- Random input is used to test the system's reliability and performance.
- Saves time and effort than actual test efforts.

Random Testing Steps:

1. Random Inputs are identified to be evaluated against the system.
2. Test Inputs are selected independently from test domain.
3. Tests are Executed using those random inputs.
4. Record the results and compare against the expected outcomes.
5. Reproduce/Replicate the issue and raise defects, fix and retest.

Requirements based testing

- Requirements-based testing is a testing approach in which test cases, conditions and data are derived from requirements. It includes functional tests and also non-functional attributes such as performance, reliability or usability.

Stages in Requirements based Testing:

- **Defining Test Completion Criteria -** Testing is completed only when all the functional and non-functional testing is complete.
- **Design Test Cases -** A Test case has five parameters namely the initial state or precondition, data setup, the inputs, expected outcomes and actual outcomes.
- **Execute Tests** - Execute the test cases against the system under test and document the results.
- **Verify Test Results -** Verify if the expected and actual results match each other.
- **Verify Test Coverage -** Verify if the tests cover both functional and non-functional aspects of the requirement.
- **Track and Manage Defects -** Any defects detected during the testing process goes through the defect life cycle and are tracked to resolution. Defect Statistics are maintained which will give us the overall status of the project.

Requirements testing process:

- Testing must be carried out in a timely manner.
- Testing process should add value to the software life cycle; hence it needs to be effective.
- Testing the system exhaustively is impossible hence the testing process needs to be efficient as well.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Using-black-box-approach-to-test-case-design_9148/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 (Page No:73-81)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE

(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)

Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 06

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 28.08.2021

Topic of Lecture: Boundary Value Analysis, Decision tables, Equivalence Class Partitioning, State-based testing

Introduction: Software Testing is imperative for a bug-free application; this can be done manually or even automated. Although automation testing reduces the testing time, manual testing continues to be the most popular method for validating the functionality of software applications.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Testing Approaches
- Black Box Testing
- Data Structure
- Memory Organization

Detailed content of the Lecture:

Boundary Value Analysis

Boundary value analysis is a type of black box or specification-based testing technique in which tests are performed using the boundary values. For example: If we want to test a field which should accept only amount more than 10 and less than 20 then we take the boundaries as 10-1, 10, 10+1, 20-1, 20, 20+1. Instead of using lots of test data, we just use 9, 10, 11, 19, 20 and 21.

Guidelines

1) Range
2) Maximum and Minimum numbers
3) Output Conditions

4) Internal Data Structures

Decision Table

Decision Table is aka Cause-Effect Table. This test technique is appropriate for functionalities which has logical relationships between inputs (if-else logic). In Decision table technique, we deal with combinations of inputs. To identify the test cases with decision table, we consider conditions and actions. We take conditions as inputs and actions as outputs.

Steps in Writing Decision Tables

- Define the problem accurately that has to be solved by a computer.
- List out all the conditions to be tested in the problem.
- List out the corresponding actions that should be taken with each combination of Conditions.

Form a decision table using the two lists.

Conditions	**R1**	**R2**	**R3**
Withdrawal Amount <= Balance	T	F	F
Credit granted	-	T	F
Actions			
Withdrawal granted	T	T	F

Advantages of Decision Tables

- When the condition are numerous, then the decision table help to visualize the outcomes of a situation
- Decision tables summarize all the outcomes of a situation & suggest suitable actions
- They provide more compact documentation
- Decision tables can be changes easily.
- Decision table have a standard format.

Equivalence Class Partioning

In this method, the input domain data is divided into different equivalence data classes. This method is typically used **to reduce the total number of test case**s to a finite set of testable test cases, still covering maximum requirements.

Test Cases

- Define the equivalence classes
- Write the initial test case
- Continue Writing test cases until all of the valid equivalence classes have been included

Guidelines

- Range
- Specific Value
- Member of a set
- Boolean

Advantages of Equivalence Partitioning

- Less number of tests
- Can cover all possible sets of tests data
- Less time in execution
- Minimizes sets of test data

Disadvantages of Equivalence Partitioning

- It does not consider the conditions for boundary value.
- The identification of equivalence classes relies heavily on the expertise of the testers

Video Content / Details of website for further learning (if any):
https://celestialsys.com/blog/software-testing-boundary-value-analysis-equivalence-partitioning/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, " Software Testing – Principles and Practices", Pearson Education, 2009 (Page No:84-93)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 07

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 31.08.2021

Topic of Lecture : Effect graphing , Error guessing , Compatibility testing

Introduction: System testing of software or hardware is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirements. System testing falls within the scope of black box testing, and as such, should require no knowledge of the inner design of the code or logic.

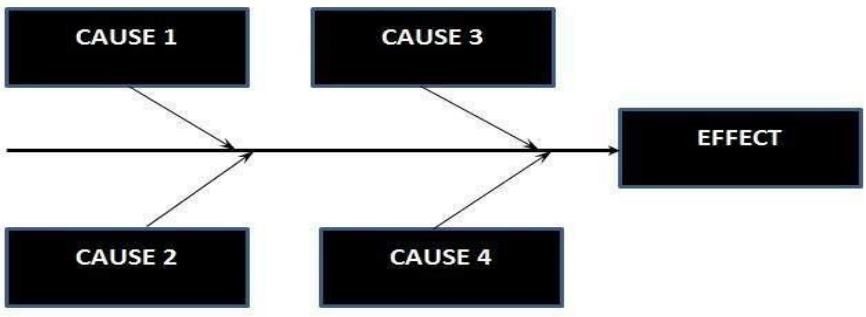
Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Testing Approaches
- Black Box Testing
- Testing Methodologies
- Graph Matrix

Detailed content of the Lecture:

Effect graphing

Cause Effect Graph is a black box testing technique that graphically illustrates the relationship between a given outcome and all the factors that influence the outcome. It is also known as Ishikawa diagram as it was invented by Kaoru Ishikawa or fish bone diagram because of the way it looks.



Steps for drawing cause-Effect Diagram

- **Step 1 :** Identify and Define the Effect
- **Step 2 :** Fill in the Effect Box and Draw the Spine
- **Step 3:** Identify the main causes contributing to the effect being studied.
- **Step 4 :** For each major branch, identify other specific factors which may be the causes of the EFFECT.
- **Step 5 :** Categorize relative causes and provide detailed levels of causes.

Advantages

- Easy to read
- It indicates possible causes of variation in a process
- Encourages team participation
- Increase the knowledge of process

Error Guessing

- Error guessing is a testing technique that makes use of a tester's skill, intuition and experience in testing similar applications to identify defects that may not be easy to capture by the more formal techniques. It is usually done after more formal techniques are completed.

Factors used to Guess the Errors

- Historical Learning
- Review Checklist
- Previous test results
- Previous defects

Advantages of Error Guessing technique

Proves to be very effective when used in combination with other formal testing techniques.

- It uncovers those defects which would otherwise be not possible to find out, through formal testing. Thus, the experience of the tester saves a lot of time and effort.
- Error guessing supplements the formal test design techniques.
- Very helpful to guess problematic areas of the application.

Disadvantages of Error Guessing technique

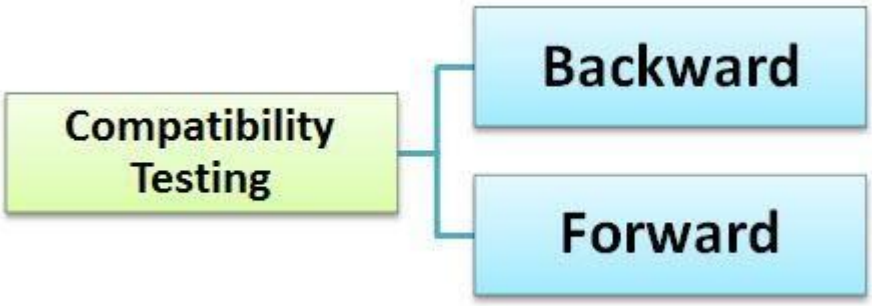
- The focal shortcoming of this technique is that it is person dependent and thus the experience of the tester controls the quality of test cases.
- It also cannot guarantee that the software has reached the expected quality benchmark.
- Only experienced testers can perform this testing. You can’t get it done by freshers.

Compatibility Testing

- Compatibility Testing is a type of Software testing to check whether you software is capable of running on different hardware, operating systems, applications, network environments or Mobile devices.

Different types of Compatibility Test

Hardware, OS, Software, Network, Devices, Mobiles, etc



Advantages

- It helps to detect errors in the software product before it is delivered to the end users.
- Reduces the future help desk cost, which is mainly incurred for providing relevant and required customer support for various compatibility issues.
- Helps test the product's scalability, stability, and usability.

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/cause-and-effect-graph-test-case-writing-technique/

Important Books/Journals for further learning including the page nos.:

Srinivasan Desikan and Gopalaswamy Ramesh, " Software Testing – Principles and Practices", Pearson Education, 2009 (Page No:94-98)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 08

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 01.09.2021

Topic of Lecture User documentation testing, Domain testing

Introduction: Test documentation is documentation of artifacts created before or during the testing of software. It helps the testing team to estimate testing effort needed, test coverage, resource tracking, execution progress, etc. It is a complete suite of documents that allows you to describe and document test planning, test design, test execution, test results that are drawn from the testing activity.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Testing Approaches
- Black Box Testing
- Graph Matrix
- System maintenance

Detailed content of the Lecture:
User Documentation

- User Documentation covers all the manuals, user guides, installation guides, setup guides, read me files, software release notes, and online ...
- **Documentation testing** is part of non-functional testing of a product. It may be a type of black box testing that ensures that documentation about how to use the system matches with what the system does, providing proof that system changes and improvement have been documented.

Benefits

Satisfied users, Support, Longevity, Sales, Usability

Domain Testing

- Domain testing is a type of functional testing and tests the application by feeding interesting inputs and evaluating its outputs. One of the most important White Box Testing method is a domain testing. The main goal of the Domain testing is to check whether the system accepts the input within the acceptable range and delivers the required output.

Skills Needed

- Domain Knowledge, Testing Skill, Technical Skill, Communication Skill, etc

Advantages of Black Box Testing

- Efficient when used on large systems.
- Since the tester and developer are independent of each other, testing is balanced and unprejudiced.
- Tester can be non-technical.
- There is no need for the tester to have detailed functional knowledge of system.
- Testing helps to identify vagueness and contradictions in functional specifications.
- Test cases can be designed as soon as the functional specifications are complete.

Disadvantages of Black Box Testing

- Test cases are challenging to design without having clear functional specifications.
- It is difficult to identify tricky inputs if the test cases are not developed based on specifications.
- It is difficult to identify all possible inputs in limited testing time.
- Large sample space is required for test inputs.
- There is a high probability of repeating tests already performed by the programmer.

Video Content / Details of website for further learning (if any):
https://www.guru99.com/testing-documentation.html

Important Books/Journals for further learning including the page nos.:

Srinivasan Desikan and Gopalaswamy Ramesh, " Software Testing – Principles and Practices", Pearson Education, 2009 **Page No: (101-105)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 09

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : I - Testing Techniques & Test Case Design Date of Lecture: 03.09.2021

Topic of Lecture: Case study for Control Flow Graph and State-based Testing.

Introduction: To illustrate weakness of decision coverage when the decision is a composite multi-condition decision.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- State Based Testing
- Testing Approaches
- Black Box Testing
- Multiple Condition Coverage
- Flow Graph

Detailed content of the Lecture:

Coverage and Control Flow Graphs

- The application of coverage analysis is typically associated with the use of control and data flow models to represent program structural elements and data.
- The logic elements most commonly considered for coverage are based on the flow of control in a unit of code. For example,

(i) program statements;

(ii) decisions/branches (these influence the program flow of control);

(iii) conditions (expressions that evaluate to true/false, and do not contain any other true/false-valued expressions);

(iv)combinations of decisions and conditions;

(v) paths (node sequences in flow graphs).

These logical elements are rooted in the concept of a program prime. A program prime is an atomic programming unit. All structured programs can be built from three basic primes-sequential (e.g., assignment statements), decision (e.g., if/then/else statements), and iterative (e.g., while, for loops)

- Using the concept of a prime and the ability to use combinations of primes to develop structured code, a (control) flow diagram for the soft- ware unit under test can be developed.
- The flow graph can be used by the tester to evaluate the code with respect to its testability, as well as to develop white box test cases. This will be shown in subsequent sections of this chapter. Note that in the flow graph the nodes represent sequential statements, as well as decision and looping predicates.
- For simplicity, sequential statements are often omitted or combined as a block that indicates that if the first statement in the block is executed, so are all the following statements in the block. Edges in the graph represent transfer of control.
- The direction of the transfer depends on the outcome of the condition in the predicate (true or false).
- There are commercial tools that will generate control flow graphs from code and in some cases from pseudo code.
- The tester can use tool support for developing control flow graphs especially for complex pieces of code.
- A control flow representation for the software under test facilitates the design of white box-based test cases as it clearly shows the logic elements needed to design the test cases using the coverage criterion of choice.

This method will presents control-flow, or logic-based, coverage concepts in a less formal but practical manner to aid the tester in developing test data sets, setting quantifiable testing goals, measuring results, and evaluating the adequacy of the test outcome. Examples based on the logic elements listed previously will be presented. Subsequent sections will describe data flow and fault-based coverage criteria.

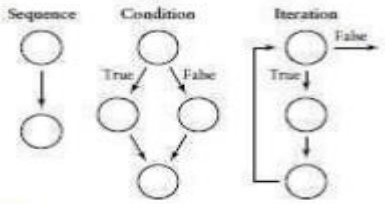


FIG. 5.1
Representation of program primes.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Coverage-and-Control-Flow-Graphs_9155/

Important Books/Journals for further learning including the page nos.:
Web Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 10

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 04.09.2021**

Topic of Lecture: The Need for Levels of Testing, Unit Test Planning, Designing the Unit Tests, The Test Harness

Introduction: Different levels of testing are used, which performs different tasks and aim to test different aspects of the system. The basic levels are unit testing, integration testing, system testing and acceptance testing. These different levels of testing attempt to detect different types of faults. The purpose of levels of testing is to make software testing systematic and easily identify all possible test cases at a particular

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Planning
- Design
- Construction
- Testing
- Testing Approaches

Detailed content of the Lecture:

The Need for Levels of Testing

- Tests are grouped together based on where they are added in SDLC or the by the level of of detailing they contain.
- In general, there are four levels of testing: unit testing, integration testing, system testing, and acceptance testing.
- The purpose of Levels of testing is to make software testing systematic and easily identify all possible test cases at a particular level.
- There are many different testing levels which help to check behavior and performance for software testing.
- These testing levels are designed to recognize missing areas and reconciliation between the development lifecycle states.
- In SDLC models there are characterized phases such as requirement gathering, analysis, design, coding or execution, testing, and deployment. All these phases go through the process of software testing levels.

Unit Test Planning

- In general unit test plan should be prepared. It may be prepared as a component of the master test plan or as a stand-alone plan. It should be developed in conjunction with the master test plan and the project plan for each project.
- Documents that provide inputs for the unit test plan are the project plan, as well the requirements, specification, and design documents that describe the target units. Components of a unit test plan are described in detail the IEEE Standard for Software Unit Testing. This standard is rich in information and is an excellent guide for the test planner.
- A brief description of a set of development phases for unit test planning is found below.
- In each phase a set of activities is assigned based on those found in the IEEE unit test standard.
- The phases allow a steady evolution of the unit test plan as more information becomes available.

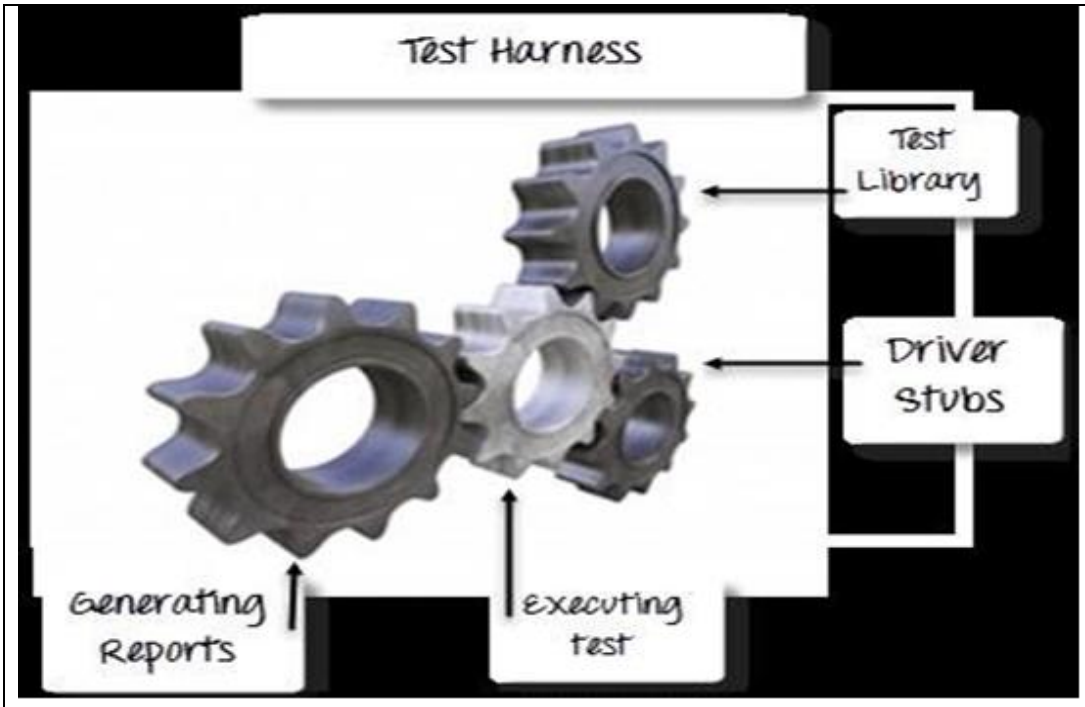
Designing the Unit Tests

- Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not.
- It is a testing method using which every independent modules are tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules.
- Unit testing of software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit testing is typically performed by the developer.

In SDLC or V Model, Unit testing is first level of testing done before integration testing. Unit testing is such type of testing technique that is usually performed by the developers. Although due to reluctance of developers to tests, quality assurance engineers also do unit testing.

The Test Harness

- Test Harness in Software Testing is a collection of stubs, drivers and other supporting tools required to automate test execution.
- Test harness executes tests by using a test library and generates test reports. Test harness contains all the information needed to compile and run a test like test cases, target deployment port (TDP), source file under test, stubs, etc.



Advantages of Test Harness

- Increased productivity due to automation of the testing process.
- Increased probability that regression testing will occur.
- Increased quality of software components and application.
- Repeatability of subsequent test runs.
- Offline testing (e.g. at times that the office is not staffed, like overnight).

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Unit-test-planning_9162/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 (Page No:106-107)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 11

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 07.09.2021**

Topic of Lecture: Running the Unit tests and Recording Results, Integration Tests, Designing Integration Tests

Introduction: Integration Testing is a level of software testing where individual units / components are combined and tested as a group. The purpose of this level of testing is to expose faults in the interaction between integrated units. Test drivers and test stubs are used to assist in Integration Testing.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Planning
- Interface
- Methods
- Function
- Testing
- Testing Approaches

Detailed content of the Lecture:

Running the Unit tests and Recording Results

- Unit tests can begin when
  (i) the units become available from the developers (an estimation of availability is part of the test plan)
  (ii) the test cases have been designed and reviewed, and
  (iii) the test harness, and any other supplemental supporting tools, are available. The testers then proceed to run the tests and record results.
- The status of the test efforts for a unit, and a summary of the test results, could be recorded in a simple format such as shown.
- These forms can be included in the test summary report, and are of value at the weekly status meetings that are often used to monitor test progress. It is very important for the tester at any level of testing to carefully record, review, and check test results.
- The tester must determine from the results whether the unit has passed or failed the test. If the test is failed, the nature of the problem should be recorded in what is sometimes called a test incident report.

- The test set will have to be augmented and the test plan documents should reflect these changes.
- When a unit fails a test there may be several reasons for the failure. The most likely reason for the failure is a fault in the unit implementation (the code).
- Other likely causes that need to be carefully investigated by the tester are the following:

• a fault in the test case specification (the input or the output was not specified correctly);

• a fault in test procedure execution (the test should be rerun);

• a fault in the test environment (perhaps a database was not set up properly);

• a fault in the unit design (the code correctly adheres to the design specification, but the latter is incorrect).

Unit Test Worksheet

Unit Name: _____

Unit Identifier: _____

Tester: _____

Date: _____

Test case ID	Status (run/not run)	Summary of results	Pass/fail

TABLE 6.1
Summary work sheet for unit test results.

Integration Tests

Integration Testing is a level of software testing 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.

- Test drivers and test stubs are used to assist in Integration Testing.

Steps of Integration Testing

Integration Testing Steps:

- Prepare Integration Test Plan.
- Prepare integration test scenarios & test cases.
- Prepare test automation scripts.
- Execute test cases.
- Report the defects.
- Track and re-test the defects.
- Re-testing & testing goes on until integration testing is complete.

Designing Integration Tests

- Integration tests for procedural software can be designed using a black or white box approach. Both are recommended. Some unit tests can be reused. Since many errors occur at module interfaces, test designers need to focus on exercising all input/output parameter pairs, and all calling relationships.
- The tester needs to insure the parameters are of the correct type and in the correct order. The author has had the personal experience of spending many hours trying to locate a fault that was due to an incorrect ordering of parameters in the calling routine.
- The tester must also insure that once the parameters are passed to a routine they are used correctly. For example, in Figure 6.9, Procedure_b is being integrated with Procedure_a. Procedure_a calls Procedure_b with two input parameters in3, in4. Procedure_b uses those parameters and then returns a value for the output parameter out1.

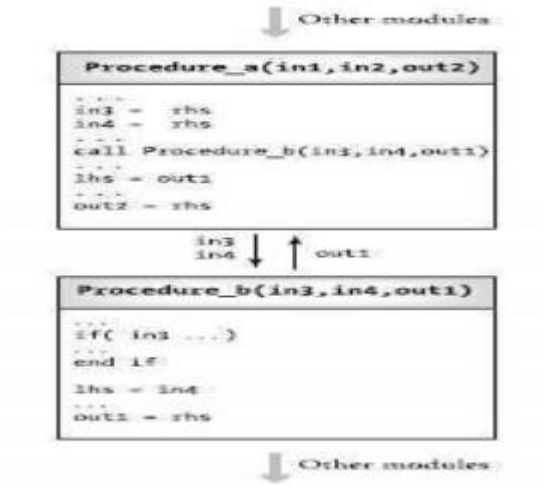


Fig. 6.9
Example integration of two procedures.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Running-the-unit-tests-and-recording-results_9166/
http://www.brainkart.com/article/Designing-integration-tests_9168/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:107-115)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 12

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 08.09.2021**

Topic of Lecture: Integration Test Planning , Scenario Testing, Defect Bash Elimination, System Testing

Introduction: System Integration Testing is defined as a type of software testing carried out in an integrated hardware and software environment to verify the behavior of the complete system. It is testing conducted on a complete, integrated system to evaluate the system's compliance with its specified requirement.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Prototypes
- Design
- Defect
- Testing
- Testing Approaches

Detailed content of the Lecture:

Integration Test Planning

- Planning can begin when high-level design is complete so that the system architecture is defined.
- Other documents relevant to integration test planning are the requirements document, the user manual, and usage scenarios.
- These documents contain structure charts, state charts, data dictionaries, cross-reference tables, module interface descriptions, data flow descriptions, messages and event descriptions, all necessary to plan integration tests.

Scenario Testing

- Scenario Testing is a Software Testing Technique that uses scenarios i.e. speculative stories to help the tester work through a complicated problem or test system.

- In scenario testing, the testers assume themselves to be the end users and find the real world scenarios or use cases which can be carried out on the software by the end user. In scenario testing, the testers take help from clients, stakeholders and developers to create test scenarios.

Defect Bash Elimination

- Defect bash is an ad hoc testing, done by people performing different roles to bring out all types of defects.
- It is very popular among application development companies, where the products can be used by people who perform different roles.
- The testing by all the participants during the defect bash is not based on written test cases.

It involves several steps:-

- Choosing the frequency and duration of defect bash.
- Selecting the right product build.
- Communicating the objectives of each defect bash to everyone
- Setting up and monitoring the lab for defect bash.
- Taking action and fixing issues.
- Optimizing the effort involved in defect bashes.

System Testing

- System Testing is a series of different tests whose primary purpose is to fully exercise the computer-based system. System Testing tests a completely integrated system to verify that it meets its requirements.

Therefore, some of the critical considerations for System testing are:

- Firstly, the performance of System testing happens in a fully developed and integrated system.
- Secondly, the performance of System tests happens on the entire system in the context of the functional requirements specifications (FRS) or the system requirements specifications (SRS), or both. In other words, System tests validate not only the design but also the behavior and usability aspects.
- In addition to the above, it verifies the entire product, after integrating all software and hardware components and validating it according to the specifications.
- Moreover, System testing can include both functional and non-functional types of testing.

Video Content / Details of website for further learning (if any):
https://www.toolsqa.com/software-testing/istqb/system-testing/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:116-132)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 13

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 14.09.2021**

Topic of Lecture: Acceptance testing , Performance testing

Introduction: Acceptance testing, a testing technique performed to determine whether or not the software system has met the requirement specifications. The main purpose of this test is to evaluate the system's compliance with the business requirements and verify if it is has met the required criteria for delivery to end users.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- End User
- Design
- Report
- Manual
- Testing Approaches

Detailed content of the Lecture:

Acceptance testing

- When software is being developed for a specific client, acceptance tests are carried out after system testing.
- The acceptance tests must be planned carefully with input from the client/users. Acceptance test cases are based on requirements.
- Clients should be provided with documents and other material to help them participate in the acceptance testing process, and to evaluate the results.
- After acceptance testing the client will point out to the developers which requirement have/have not been satisfied.
- If the client is satisfied that the software is usable and reliable, and they give their approval, then the next step is to install the system at the client's site.

Performance testing

- Performance Testing also knows as 'Perf Testing', is a type of testing performed to check how application or software performs under workload in terms of responsiveness and stability.
- The Performance Test goal is to identify and remove performance bottlenecks from an application
- This test is mainly performed to check whether the software meets the expected requirements for application speed, scalability, and stability.

Reasons for Performance Testing

Organizations run performance testing for at least one of the following reasons:

- To determine whether the application satisfies performance requirements (for instance, the system should handle up to 1,000 concurrent users).
- To locate computing bottlenecks within an application.
- To establish whether the performance levels claimed by a software vendor are indeed true.
- To compare two or more systems and identify the one that performs best.
- To measure stability under peak traffic events.

Types of Performance Testing

Load testing
Stress testing
Spike testing
Endurance testing
Scalability testing
Volume testing

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/introduction-to-performance-testing-loadrunner-training-tutorial-part-1/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:160-192)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 14

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 15.09.2021**

Topic of Lecture: Regression Testing , Internationalization testing , Ad-hoc testing

Introduction: Testing to verify a product meets customer specified requirements. This is a non-methodical approach where testing is performed, in general, without planning and documentation. Here the tester tries to 'break' the system by randomly trying the system's functionality. This includes negative testing as well.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Planning
- Standards
- Quality
- Multiplatform Independence
- Testing Approaches

Detailed content of the Lecture:

Regression Testing

- Regression testing a black box testing technique that consists of re-executing those tests that are impacted by the code changes. These tests should be executed as often as possible throughout the software development life cycle.

Types of Regression Tests:

- **Final Regression Tests: -** A "final regression testing" is performed to validate the build that hasn't changed for a period of time. This build is deployed or shipped to customers.
- **Regression Tests: -** A normal regression testing

Internationalization testing

- Internationalization testing is the process of verifying the application under test to work uniformly across multiple regions and cultures.
- The main purpose of internationalization is to check if the code can handle all international support without breaking functionality that might cause data loss or data integrity issues. Globalization testing verifies if there is proper functionality of the product with any of the locale settings.

Internationalization Checklists:

- Testing to check if the product works across settings.
- Verifying the installation using various settings.
- Verify if the product works across language settings and currency settings.

Ad-hoc testing

- When a software testing performed without proper planning and documentation, it is said to be Adhoc Testing. Such kind of tests are executed only once unless we uncover the defects.
- Adhoc Tests are done after formal testing is performed on the application. Adhoc methods are the least formal type of testing as it is NOT a structured approach. Hence, defects found using this method are hard to replicate as there are no test cases aligned for those scenarios.
- Testing is carried out with the knowledge of the tester about the application and the tester tests randomly without following the specifications/requirements.

Forms of Adhoc Testing :

1. **Buddy Testing:** Two buddies, one from development team and one from test team mutually work on identifying defects in the same module. Buddy testing helps the testers develop better test cases while development team can also make design changes early. This kind of testing happens usually after completing the unit testing.
2. **Pair Testing:** Two testers are assigned the same modules and they share ideas and work on the same systems to find defects. One tester executes the tests while another tester records the notes on their findings.
3. **Monkey Testing:** Testing is performed randomly without any test cases in order to break the system.

Various ways to make Adhoc Testing More Effective

1. **Preparation:** By getting the defect details of a similar application, the probability of finding defects in the application is more.
2. **Creating a Rough Idea:** By creating a rough idea in place the tester will have a focussed approach. It is NOT required to document a detailed plan as what to test and how to test.
3. **Divide and Rule:** By testing the application part by part, we will have a better focus and better understanding of the problems if any.
4. **Targeting Critical Functionalities:** A tester should target those areas that are NOT covered while designing test cases.
5. **Using Tools:** Defects can also be brought to the lime light by using profilers, debuggers and even task monitors. Hence being proficient in using these tools one can uncover several defects.

Video Content / Details of website for further learning (if any):
https://www.tutorialspoint.com/software_testing_dictionary/adhoc_testing.htm

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:193-248)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 15

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 17.09.2021**

Topic of Lecture: Alpha, Beta Tests, Testing OO systems

Introduction: **Alpha and Beta testing** are the Customer Validation methodologies (Acceptance Testing types) that help in building confidence to launch the product, and thereby results in the success of the product in the market. Testing is a continuous activity during software development. In object-oriented systems, testing encompasses three levels, namely, unit testing, subsystem testing, and system testing.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Tester
- Object
- Class
- Interface
- Testing Approaches

Detailed content of the Lecture:

Alpha Testing

- Alpha Testing is a type of software testing performed to identify bugs before releasing the product to real users or to the public. Alpha Testing is one of the user acceptance testings.
- Alpha Testing can be defined as a form of acceptance testing carried out to identify various types of issues or bugs before publishing the build or executable of the software public or market. This test type focuses on real users through black box and white box testing techniques. The focus remains on the task which a general user might want or experience.
- Alpha testing any product is done when product development is on the verge of completion. Slight changes in design can be made after conducting the alpha test. This testing methodology is performed in lab surroundings by the developers.
- Here developers see things in the software from users' points and try to detect the problems. These testers are internal company or organization's employees or maybe a part of the testing team.
- Alpha testing is done early at the end of software development before beta testing.

Beta Testing

- Beta Testing is performed by real users of the software application in a real environment. Beta testing is one of the types of User Acceptance Testing.
- Beta Testing can be defined as the second stage of testing any product before release, where a sample of the released product with minimum features and characteristics is being given to the intended audience for trying out or temporarily using the product.
- Unlike an alpha test, the beta test is being carried out by real users in the real environment. This allows the targeted customers to dive into the product's design, working, interface, functionality, etc.

Testing OO systems

- Testing is a continuous activity during software development. In object- oriented systems, testing encompasses three levels, namely, unit testing, subsystem testing, and system testing.
- Software typically undergoes many levels of testing, from unit testing to system or acceptance testing. Typically, in-unit testing, small "units", or modules of the software, are tested separately with focus on testing the code of that module.
- In higher, order testing (e.g, acceptance testing), the entire system (or a subsystem) is tested with the focus on testing the functionality or external behavior of the system.
- As information systems are becoming more complex, the object-oriented paradigm is gaining popularity because of its benefits in analysis, design, and coding.
- Conventional testing methods cannot be applied for testing classes because of problems involved in testing classes, abstract classes, inheritance, dynamic binding, message, passing, polymorphism, concurrency, etc.

Unit Testing

- In unit testing, the individual classes are tested.

Subsystem Testing

- This involves testing a particular module or a subsystem and is the responsibility of the subsystem lead.

System Testing

- System testing involves testing the system as a whole and is the responsibility of the quality-assurance team.

Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/object-oriented-testing-in-software-testing/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:137-140)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 16

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 18.09.2021**

Topic of Lecture: Usability and Accessibility Testing, Configuration Testing

Introduction: Accessibility testing is a subset of usability testing where in the users under consideration people with all abilities and disabilities are. The significance of this testing is to verify both usability and accessibility.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- System
- End User
- Hardware
- Testing
- Testing Approaches

Detailed content of the Lecture:

Usability Testing

- The testing that validates the ease of use, speed, and aesthetics of the product from the users' point of view is called usability testing.

Usability testing can be done is two phases:

1. Design validation Phase

2. Component and integration testing phase. Usability design is verified through several means. Some of them are as follows:

a. Style sheets b. Screen prototypes c. Paper designs d. Layout Design.

Usability Testing Process

- Planning
- Recruiting
- Usability Testing
- Data Analysis
- Reporting

Accessibility Testing

Verifying the product usability for physically challenged users is called accessibility testing.

Accessibility to the product can be provided by two means:

a. Making use of accessibility features provided by the underlying infrastructure called basic accessibility

b. Providing accessibility in the product through standards and guidelines, called product accessibility

Examples

- Speech Recognition Software
- Screen Reader Software
- Special keyboard

Configuration Testing

Configuration testing is the process of testing the system with each one of the supported software and hardware configurations. The Execution area supports configuration testing by allowing reuse of the created tests.

Executing Tests with Various Configurations:

Operating System Configuration - Win XP, Win 7 32 bit/64 bit, Win 8 32 bit/64 bit

Database Configuration - Oracle, DB2, MySql, MSSQL Server, Sybase

Browser Configuration - IE 8, IE 9, FF 16.0, Chrome

Video Content / Details of website for further learning (if any):
https://www.invensis.net/it-outsourcing-services/software-testing/outsource-usability-accessibility-testing-services

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009 **(Page No:274-295)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 17

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 21.09.2021**

Topic of Lecture: Compatibility Testing, Testing the documentation, Website Testing

Introduction: Compatibility is non-functional testing to ensure customer satisfaction. It is to determine whether your software application or product is proficient enough to run in different browsers, database, hardware, operating system, mobile devices, and networks.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Planning
- Browser
- Documentation
- Web Technology
- Client/Server
- Testing Approaches

Detailed content of the Lecture:

Compatibility Testing

- Compatibility tests ensures that your web application displays correctly across different devices
- Compatibility testing is non-functional testing. It is carried out to verify whether an application or software can run on different devices, browsers, operating systems, hardware, and networks.

Example

Browser Compatibility Test: Same website in different browsers will display differently.

Types of Compatibility Testing

- Backward Compatibility Testing
- Forward Compatibility Testing

General process for Compatibility Testing

- Platform/Environment Identification
- Design Test Cases & Configuration
- Establish Test Cases & Environment
- Results Analysis & Reporting
- Rectification & Retesting

Advantages

- Apart from the reduced cost of customer service, compatibility testing also helps in attracting and retaining more customers as it is directly connected to giving customer satisfaction, which helps in increasing an organization's revenue.
- Test documentation helps one to improve transparency with the client
- This testing ensures that the application works on all the possible combinations of devices, browsers, OS, etc. as desired.

Testing the Documentation

- Documentation testing is a non-functional type of software testing.
- Testing documentation is usually associated with the documentation artifacts that should be developed before testing of software.
- It helps the testing team to estimate testing effort needed, test coverage, resource tracking, etc

Advantages

- Test documentation is to either reduce or remove any uncertainties about the testing activities.
- Helps you to remove ambiguity which often arises when it comes to the allocation of tasks.
- Test documentation helps you to improve transparency with the client

Disadvantages

- The cost of the documentation may surpass its value as it is very time consuming
- Keeping track of changes requested by the client and updating corresponding documents is tiring.
- Poor documentation directly reflects the quality of the product as a misunderstanding between the client and the organization can occur

Website Testing/Web Application Testing

- Web testing is a software testing practice to test websites or web applications for potential bugs. It's a complete testing of web-based applications before making live.
- A web-based system needs to be checked completely from end-to-end before it goes live for end users.

Testing methods for Web Application Testing

- Functional Testing
- Usability Testing
- Interface Testing
- Compatibility Testing
- Performance Testing
- Security Testing

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/software-compatibility-testing/

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing. **(Page No:789-801)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 18

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : II - Levels of Testing **Date of Lecture: 22.09.2021**

Topic of Lecture: Case Study for Unit and Integration Testing.

Introduction: A function written as a part of the code may be invoked by multiple classes and testing functions require understanding of the class hierarchy, unlike in the traditional programming paradigm.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Requirements
- Planning
- Design
- Interaction
- Interface
- Testing Approaches

Detailed content of the Lecture:

Case Study for Unit and Integration Testing.

Unit Testing

- Unit Tests are conducted by developers and test the unit of code(aka module, component) he or she developed.
- It is a testing method by which individual units of source code are tested to determine if they are ready to use.
- It helps to reduce the cost of bug fixes since the bugs are identified during the early phases of the development lifecycle.

Integration Testing

- Integration testing is executed by testers and tests integration between software modules. It is a software testing technique where individual units of a program are combined and tested as a group.
- Test stubs and test drivers are used to assist in Integration Testing.
- Integration test is performed in two way, they are a bottom-up method and the top-down method.

Unit Test Planning

- In general unit test plan should be prepared. It may be prepared as a component of the master test plan or as a stand-alone plan. It should be developed in conjunction with the master test plan and the project plan for each project.
- Documents that provide inputs for the unit test plan are the project plan, as well the requirements, specification, and design documents that describe the target units.
- Components of a unit test plan are described in detail the IEEE Standard for Software Unit Testing. This standard is rich in information and is an excellent guide for the test planner.
- A brief description of a set of development phases for unit test planning is found below. In each phase a set of activities is assigned based on those found in the IEEE unit test standard
- The phases allow a steady evolution of the unit test plan as more information becomes available.

Designing the Unit Tests

- Unit Testing is a software testing technique by means of which individual units of software i.e. group of computer program modules, usage procedures and operating procedures are tested to determine whether they are suitable for use or not.
- It is a testing method using which every independent modules are tested to determine if there are any issue by the developer himself. It is correlated with functional correctness of the independent modules.
- Unit testing is defined as a type of software testing where individual components of a software are tested.
- Unit testing of software product is carried out during the development of an application. An individual component may be either an individual function or a procedure. Unit testing is typically performed by the developer.
- In SDLC or V Model, Unit testing is first level of testing done before integration testing.
- Unit testing is such type of testing technique that is usually performed by the developers.
- Although due to reluctance of developers to tests, quality assurance engineers also do unit testing.

Video Content / Details of website for further learning (if any):
https://www.guru99.com/unit-test-vs-integration-test.html

Important Books/Journals for further learning including the page nos.:
Web Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 19

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 24.09.2021

Topic of Lecture: Testing Client / Server Systems, Testing in a Multiplatform Environment

Introduction: Client-server is software architecture consists of client and server systems which communicate to each other either over the computer network or on the same machine. In Client-Server Application Testing, the client system sends the request to the server system and the server system sends the response to the client system. This is also known as a two-tier application.

Prerequisite knowledge for Complete understanding and learning of Topic:

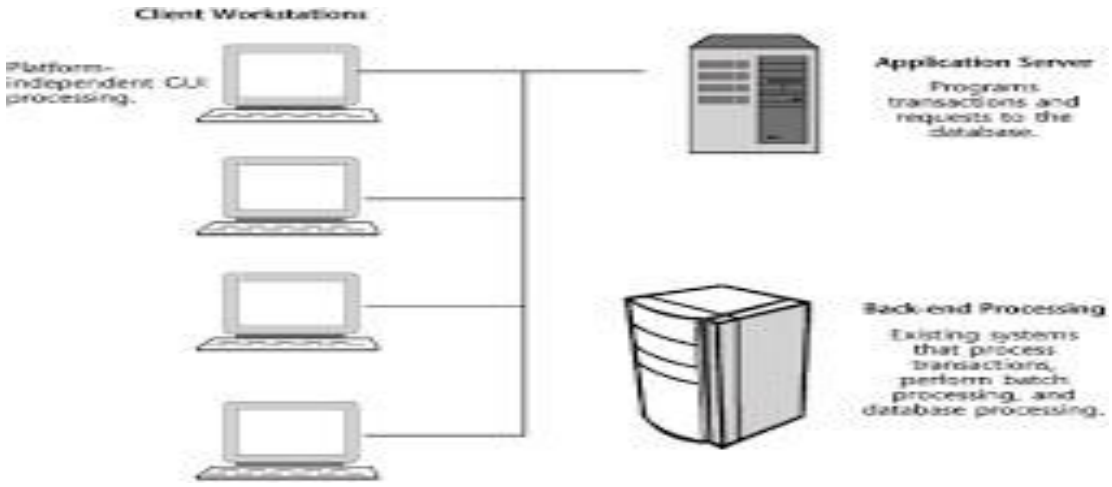
- Software
- Client
- Server
- Middleware
- Network
- Operating System
- Testing Approaches

Detailed content of the Lecture:

Testing Client / Server Systems

- The client-server model of computing is a distributed application assembly that divide tasks or capabilities between workers of a resource or facility, called servers, and service supplicants, called clients.
- Client and server architecture are a computing model in which the server hosts, delivers and manages most of the resources, request and services to be expended by the client.
- This type of architecture has one or more clients are associated to a central server by the network or Internet linking.

Diagram – Client-Server Architecture



Client-Server - Advantages

- Improved Data Sharing □□Integration of Services
- Shared Resources Amongst Different Platforms
- Easy Maintenance
- Security

Client- Server- Disadvantages

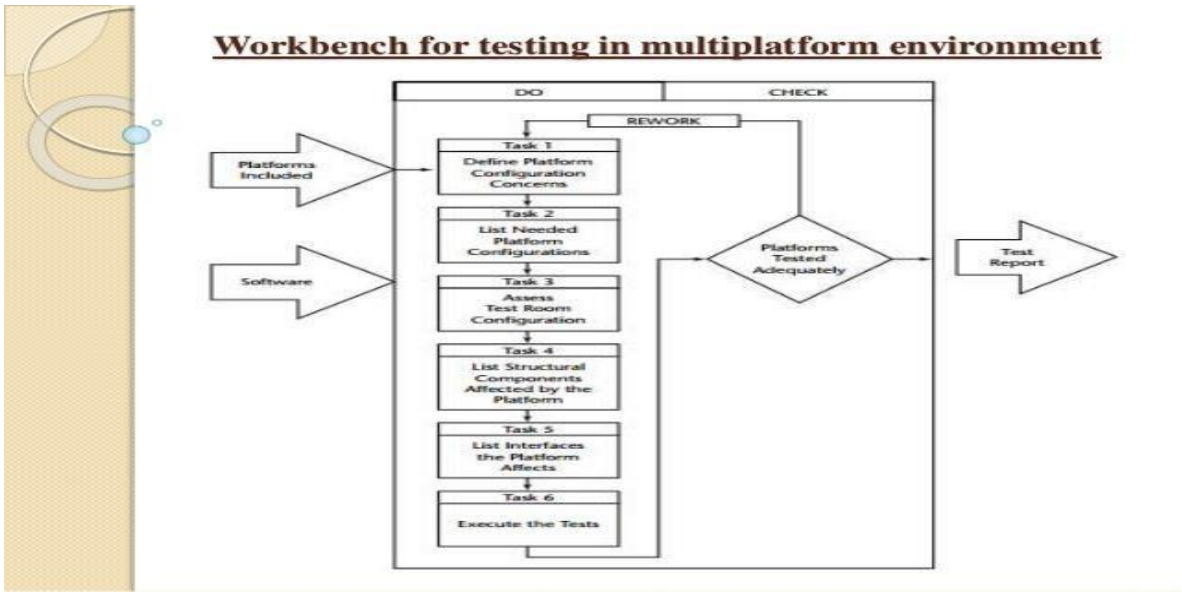
- Overloaded Servers
- Impact of Centralized Architecture
- Congestion in Network
- Expensive

Testing in Multiplatform Environment

- Each platform on which software is designed to execute operationally may have slightly different characteristics.
- These distinct characteristics include various operating systems, hardware configurations, operating instructions, and supporting software, such as database management systems.
- These different characteristics may or may not cause the software to perform its intended functions differently. The objective of testing is to determine whether the software will produce the correct results on various platforms.

Challenges Faced by Testers in Multiplatform Environment

- Determining the type of platform that users operate for the processing
- Determining which software packages are available to those users
- Determining the type of processing users used in user environment



Video Content / Details of website for further learning (if any):
http://www.softwaretestingstuff.com/2008/04/testing-client-server-applications.html

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing. **(Page No:615-625)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 20

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 28.09.2021

Topic of Lecture: Testing Object-Oriented Software, Object Oriented Testing

Introduction: Object-oriented programming increases software reusability, extensibility, interoperability, and reliability. Software testing is necessary to realize these benefits by uncovering as many programming errors as possible at a minimum cost. A major challenge to the software engineering community remains how to reduce the cost while improving the quality of software testing.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing Approaches
- Object
- Class
- Inheritance
- Polymorphism
- Encapsulation

Detailed content of the Lecture:

Testing Object-Oriented Software

- Object-oriented programming features in programming languages obviously impact some aspects of testing. Features such as class inheritance and interfaces support polymorphism in which code manipulate objects without their extract class being known

Issues- OO Testing

- Basic unit of unit testing.
- Implication of Encapsulation.
- Implication of Inheritance.
- Implication of Genericity.

- In OO paradigm, software engineers identify and specify the objects and services provided by each object.
- The main advantages of OO paradigm include reusability, reliability, interoperability and extendibility. It should be tested at different levels to uncover all the errors.
- White box testing can be applied easily

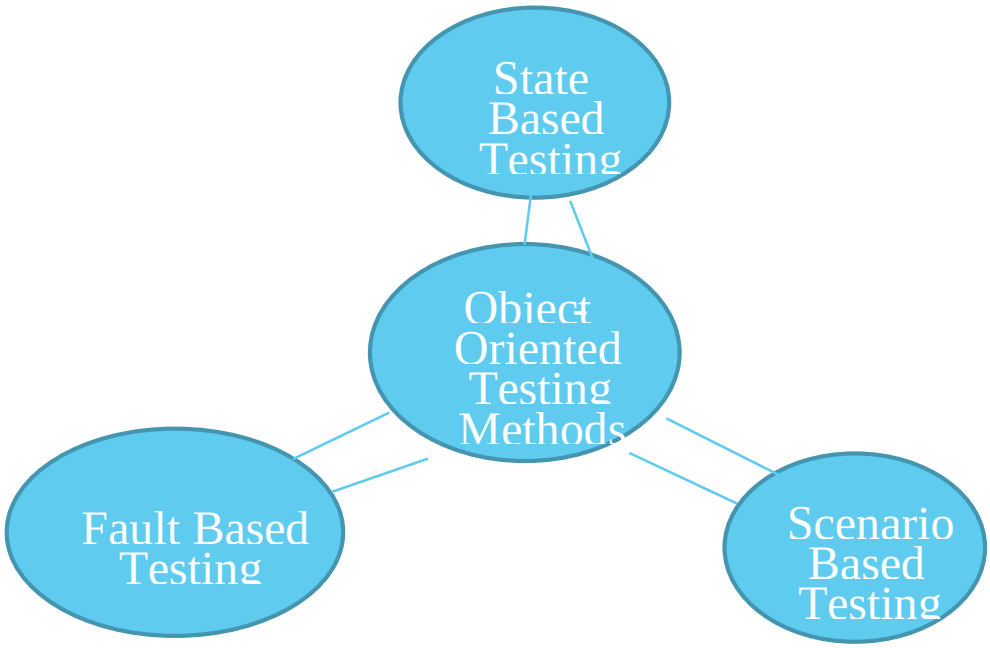
Features of OO Testing

- To perform effective testing, a software team should conduct effective formal technical reviews.
- Testing begins at the component level and work outward toward the integration of the entire computer-based system
- Different testing techniques are appropriate at different points in time
- Testing is conducted by the developer of the software and an independent test group
- Testing and debugging are different activities.

Developing Test Cases in Object-Oriented Testing

- It should be explicitly specified with each test case
- Purpose of each test case should be mentioned
- All the states of objects that is to be tested should be specified
- Instructions to understand and conduct the test cases should be provided with each test case

Object Oriented Testing Methods



State-Based Testing

- It is used to verify whether the methods of a class are interacting properly with each other.
- This testing seeks to exercise the transitions among the states of based upon the identified inputs

Fault-based Testing

- This type of checking permits for coming up with test cases supported the consumer specification or the code or both.
- It tries to identify possible faults
- For all of these faults, a test case is developed to “flush” the errors out. These tests also force each time of code to be executed.
- These types of errors can be uncovered by function testing in the traditional testing model.

Scenario-Based Testing

- It primarily involves capturing the user actions then stimulating them to similar actions throughout the test.
- These tests tend to search out interaction form of error.
- This is used to detect errors that are caused due to incorrect specifications and improper interactions among various segments of the software

Object Oriented (OO) Testing Strategies

Unit Testing in OO Context:

This means that each class and each instance of a class, attributes and operations that manipulate these data

Integration Testing in OO Context:

Integrating operations one at a time into a class is often impossible because of the "direct and indirect interactions of the components that make up the class"

Validation Testing in an OO Context:

At the validation/system level, the details of class connections disappear. Validation of OO software focuses on user-visible actions and user recognizable output from the system

Video Content / Details of website for further learning (if any):
https://www.wiley.com/en-us/Testing+Object+Oriented+Software-p-9780818685200

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing.
(Page No:721-729)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 21

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 29.09.2021

Topic of Lecture: Testing Web based systems, Web based system

Introduction: Web-based systems are those systems that use the Internet, intranets, and extranets. The Internet is a worldwide collection of interconnected networks. An intranet is a private network inside a company using web-based applications, but for use only within an organization. An extranet is a private network that allows external access to customers and suppliers using web-based applications.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing Approaches
- Internet
- Client
- Server
- Web Page
- Network

Detailed content of the Lecture:

Testing Web based systems

- Web based architecture is an extension of client/server architecture.
- For web-based systems, the browsers reside on client work stations. These client work-stations are networked to a web server, either through a remote connection or through a network such as LAN/WAN.
- Web-based systems are those systems that use the Internet, intranets, and extranets.
- The Internet is a worldwide collection of interconnected networks.
- An intranet is a private network inside a company using web-based applications, but for use only within an organization.
- An extranet is a private network that allows external access to customers and suppliers using web-based applications.

Web-based Systems

- A web-based system provides access to a software system using a computer and internet connection.
- It is a living system. Helps to create an infrastructure that will allow evolution and maintenance of a web system.

- Web systems and applications are always under continuous evolution due to technology up gradation and business dynamics

Characteristics of Web based system

- Network intensive
- Multiple user types with diverse needs
- Content driven
- Appealing and attractive aesthetics
- Use of multiple data types
- Requires IT and non-IT skill resource

Advantages

- Web based systems are more accessible
- Web based systems have a lower maintenance and deployment

Video Content / Details of website for further learning (if any):
https://www.oreilly.com/library/view/effective-methods-for/9780764598371/ch22.html

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing.
(Page No:675-689)

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 22

MCA

II / III

Course Name with Code : 19CAC21 / Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 01.10.2021

Topic of Lecture: Web Technology Evolution

Introduction: Web technologies have emerged over the times to offer web designers and developers the skills to make a completely new peer of immersive and valuable experiences on the web to the users. The web that we see today is the outcome of regular ongoing efforts of an open community of web that assists in designing the latest technologies. Some of the techs that are being used today for web development are CSS3, WebGL, HTML 5, Java, React JS, Angular JS, PHP, etc. These technologies also ensure that the website or web app is supported in all web browsers.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Testing
- Scripting
- Java
- Network
- PHP

Detailed content of the Lecture:

Web Technology Evolution

- It was introduced in 1980s.
- The development responsible for this change within the web technology landscape are spread across different fields.
- It has to do with advancements in mark-up languages, programming languages, server technology.
- Data transmission technology and infrastructure that supports internet.

Key areas of evolution of web technologies

- HTML
- Development in Browsers
- Advancement in web server technology
- Web Technology of the future

Categories of Web Based Systems

Category/Functionality	Examples
Informational	Online newspaper, newsletter, online books
Interactive	Registration forms, online games
Transactional	Online Shopping, online banking
Workflow oriented	Inventory management, Supply chain Management
Collaborative work environments	Distributed authoring systems
Online communities, marketplaces	Discussion groups,e-malls

Video Content / Details of website for further learning (if any):
https://www.decipherzone.com/blog-detail/evolution-web-development

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing. **(Page No:700-708)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 23

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 05.10.2021

Topic of Lecture: Traditional Software and Web based Software**,** Challenges in Testing for Web-based Software

Introduction: A web application is application software that runs on a web server, unlike computer-based software programs that are run locally on the operating system of the device.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- State Based Testing
- Testing Approaches
- C
- Object
- Class
- Inheritance

Detailed content of the Lecture:

Traditional Software and Web based Software

- Clients of the traditional client-server systems are platform-specific. This means the client application is developed and tested for each supported client operating system. But the web client is operating within the web browser's environment.
- Web-based systems have a more dynamic environment as compared to traditional client-server systems. In client-server systems, the roles of the clients and servers and their interactions and are predefined and static as compared to web applications where client-side programs and contents may be generated dynamically.
- In the traditional client-server systems, the normal flow of control is not affected by the user. But in web applications, users can break the normal control for example, users can press the back or refresh button in the web browser.

Challenges in Testing for Web-based software

- Integration
- Interoperability
- Security
- Performance
- Usability

Client / Server Testing

- This type of testing usually done for 2 tier applications (usually developed for LAN).The application launched on front-end will be having forms and reports which will be monitoring and manipulating data. **For Example,** applications developed in VB, VC++, Core Java, C, C++, D2K, PowerBuilder, etc., The backend for these applications would be MS Access, SQL Server, Oracle, Sybase, Mysql, Quadbase

The tests performed on these types of applications would be

- User Interface Testing
- Manual Support Testing
- Functionality Testing
- Compatibility Testing & Configuration Testing
- Intersystem Testing

Web Testing

This is done for 3 tier applications (developed for Internet / intranet / xtranet) Here we will be having Browser, web server and DB server. The applications accessible in the browser would be developed in HTML, DHTML, XML, JavaScript, etc. Applications for the webserver would be developed in Java, ASP, JSP, VBScript, JavaScript, Perl, Cold Fusion, PHP, etc.The DB server would be having Oracle, SQL Server, Sybase, MySQL, etc.

The tests performed on these types of applications would be

- User Interface Testing
- Functionality Testing
- Security Testing
- Browser Compatibility Testing
- Load/Stress Testing
- Interoperability Testing/Intersystem Testing
- Storage and Data Volume Testing

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/what-is-client-server-and-web-based-testing-and-how-to-test-these-applications/

Important Books/Journals for further learning including the page nos.:
Naresh Chauhan , "Software Testing Principles and Practices " Oxford University Press , New Delhi ,2010**(Page No: 476-477)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 24

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 06.10.2021

Topic of Lecture: Quality Aspects

Introduction: Software quality measures whether software satisfies its requirements. Software requirements are classified as either functional or non-functional.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- State Based Testing
- Testing Approaches
- Software Metrics
- Software Quality
- Requirement

Detailed content of the Lecture:

Quality Aspects

Reliability

- Measure if the product is reliable enough to sustain in any condition. Should give consistently correct results.
- Product reliability is measured in terms of working of the project under different working environments and different conditions.

Maintainability

- Different versions of the product should be easy to maintain.
- For development it should be easy to add code to the existing system, should be easy to upgrade for new features and new technologies from time to time.
- Maintenance should be cost-effective and easy.

Usability

- This can be measured in terms of ease of use. The application should be user-friendly. Should be easy to learn. Navigation should be simple.

The system must be:

- Easy to use for input preparation, operation, and interpretation of the output.
- Provide consistent user interface standards or conventions with our other frequently used systems.
- Easy for new or infrequent users to learn to use the system.

Portability

- This can be measured in terms of Costing issues related to porting, Technical issues related to porting, Behavioral issues related to porting.

Correctness

- The application should be correct in terms of its functionality, calculations used internally and the navigation should be correct. This means the application should adhere to functional requirements.

Efficiency

- Major system quality attribute. Measured in terms of time required to complete any task given to the system.

Integrity or Security

- Integrity comes with security. System integrity or security should be sufficient to prevent unauthorized access to system functions, preventing information loss, ensure that the software is protected from virus infection, and protecting the privacy of data entered into the system.

Flexibility

- Should be flexible enough to modify. Adaptable to other products with which it needs interaction. Should be easy to interface with other standard 3rd party components.

Reusability

- Software reuse is a good cost-efficient and time-saving development way. Different code library classes should be generic enough to use easily in different application modules. Dividing the application into different modules so that modules can be reused across the application.

Interoperability

- Interoperability of one system to another should be easy for the product to exchange data or services with other systems. Different system modules should work on different operating system platforms, different databases, and protocol conditions.

Video Content / Details of website for further learning (if any):

https://www.altexsoft.com/blog/engineering/what-software-quality-really-is-and-the-metrics-you-can-use-to-measure-it/

Important Books/Journals for further learning including the page nos.:

Naresh Chauhan , "Software Testing Principles and Practices " Oxford University Press , New Delhi ,2010**(Page No: 478-479)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 25

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 08.10.2021

Topic of Lecture: Web Engineering

Introduction: **Web engineering** focuses on the methodologies, techniques, and tools that are the foundation of Web application development and which support their design, development, evolution, and evaluation. Web application development has certain characteristics that make it different from traditional software, information system, or computer application development.

Prerequisite knowledge for Complete understanding and learning of Topic:

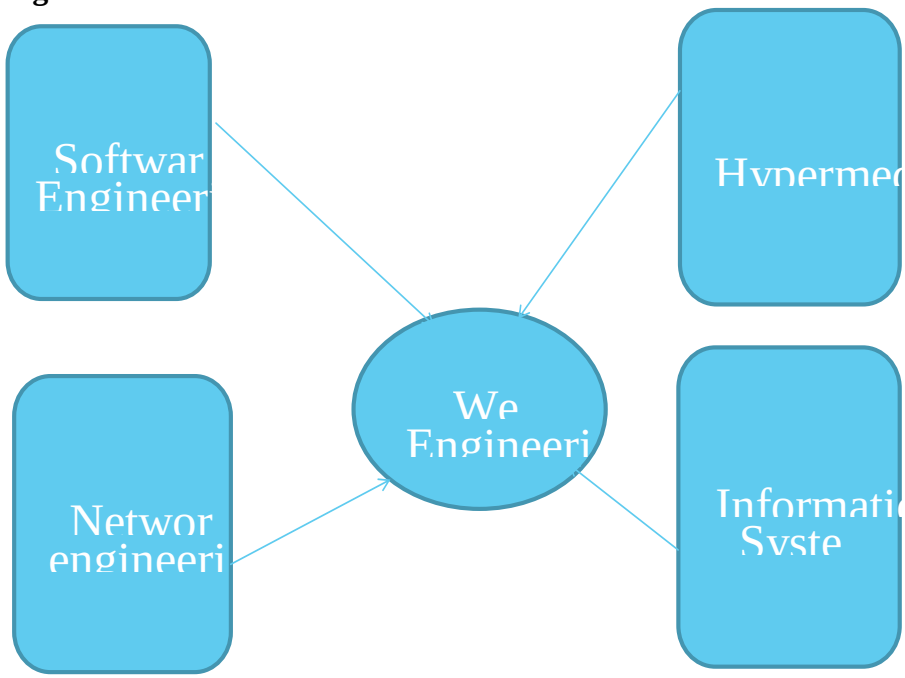
- Software
- State Based Testing
- Testing Approaches
- Web Server
- Client
- Script
- Java
- Database

Detailed content of the Lecture:

Web engineering

- Web Engineering is the application of systematic, disciplined and quantifiable approaches to development, operation, and maintenance of Web-based applications. It is both a pro-active approach and a growing collection of theoretical and empirical research in Web application development.
- It is a way of developing and organizing knowledge about web application development and applying that knowledge to develop web application
- It is a way of managing the complexity and diversity of web applications.
- Web engineering is way of developing and organizing knowledge about Web application development and applying that knowledge to develop Web applications, or to address new requirements or challenges. It is also a way of managing the complexity and diversity of Web applications.
- A Web-based system is a living system. It is like a garden — it continues to evolve, change, and grow. A sound infrastructure must be in place to support the growth of a Web-based system in a controlled, but flexible and consistent manner. Web engineering helps to create an infrastructure that will allow evolution and maintenance of a Web system and that will also support creativity. Web engineering is application of scientific, engineering, and management principles and disciplined and systematic approaches to the successful development

Web Engineering-Diagram



Need of Web Engineering

- Web Developer's Experience, New Technologies
- Characteristics and Complexity of Web Applications
- Multidisciplinary Nature of Web Development

Video Content / Details of website for further learning (if any):
http://web.nchu.edu.tw/~pfsum/WDM/Web_Engineering_Intro.pdf

Important Books/Journals for further learning including the page nos.:
Naresh Chauhan , "Software Testing Principles and Practices " Oxford University Press , New Delhi ,2010**(Page No: 480-483)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 26

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 20.10.2021

Topic of Lecture: Testing of Web based Systems

Introduction: Web-based systems are those systems that use the Internet, intranets, and extranets. The Internet is a worldwide collection of interconnected networks. An intranet is a private network inside a company using web-based applications, but for use only within an organization. An extranet is a private network that allows external access to customers and suppliers using web-based applications

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- State Based Testing
- Testing Approaches
- Client
- Server
- Network
- Middleware

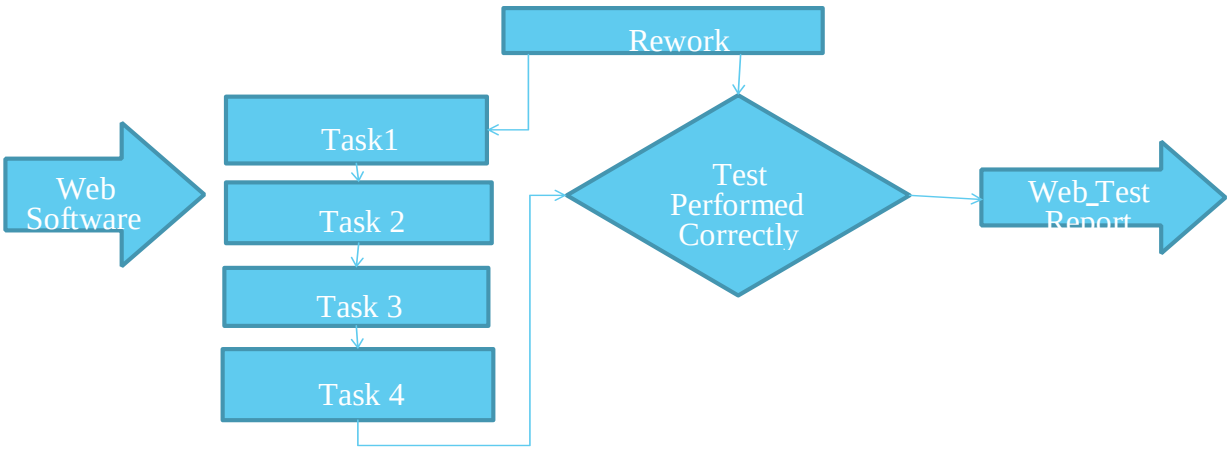
Detailed content of the Lecture:

Testing of Web based Systems

- Web testing is a software testing practice to test websites or web applications for potential bugs. It's a complete testing of web-based applications before making live.
- A web-based system needs to be checked completely from end-to-end before it goes live for end users.
- By performing website testing, an organization can make sure that the web-based system is functioning properly and can be accepted by real-time users.

Work-bench of Web-based Texting

Do **Check**



Step 1:Select Web-Based Risks to Include in the Test Plan(security, Reliability, Usability, Completeness, etc)
Step2: Select Web-Based Tests(unit, integration ,system ,user acceptance, performance, regression, etc)

Step 3:Select Web-Based Test Tools(HTML Tools, Site Validation Tools, Load/Stress Testing Tools, Test Case Generators)

Step 4: Test Web-Based Systems: This process may have to be modified based on the risks associated with web-based testing.

Concerns in Web-based Systems

- Browser Compatibility
- Function Correctness
- Integration
- Usability
- Security
- Performance
- Verification of Code

Video Content / Details of website for further learning (if any):
https://www.softwaretestingmentor.com/how-to-test-web-based-applications/

Important Books/Journals for further learning including the page nos.:
Naresh Chauhan , "Software Testing Principles and Practices " Oxford University Press , New Delhi ,2010**(Page No: 484-490)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 27

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : III - Testing For Specialized Environment Date of Lecture: 22.10.2021

Topic of Lecture: Case Study for Web Application Testing

Introduction: Software Testing is an important process to assure the quality of software including web-based application. The aim of testing is to detect all failures to ensure that the software is built in accordance with its specifications. There are two methods of testing, such as the white box and the black box testing. White box testing is done with the approach to the program code, while black box testing is done without referring to the source code, but to the output of the resulting program.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- State Based Testing
- Testing Approaches
- Client
- Server
- Network
- Java
- HTML

Detailed content of the Lecture:

Case Study for Web Application Testing

- Web testing is a software testing practice to test websites or web applications for potential bugs. It's a complete testing of web-based applications before making live.
- A web-based system needs to be checked completely from end-to-end before it goes live for end users.
- By performing website testing, an organization can make sure that the web-based system is functioning properly and can be accepted by real-time users.

The UI design and functionality are the captains of website testing.

Web Testing Checklists

1) Functionality Testing
2) Usability testing
3) Interface testing
4) Compatibility testing
5) Performance testing
6) Security testing

Cookie Testing:

- Cookies are small files stored on the user machine. These are basically used to maintain the session – mainly the login sessions.
- Test the application by enabling or disabling the cookies in your browser options.
- Test if the cookies are encrypted before writing to the user machine. If you are testing session cookies

Validate your HTML/CSS:

If you are optimizing your site for Search engines then HTML/CSS validation is the most important one. Mainly validate the site for HTML syntax errors. Check if the site is crawl able to different search engines.

Database Testing:

- Data consistency is also very important in a web application. Check for data integrity and errors while you edit, delete, modify the forms or do any DB related functionality.
- Check if all the database queries are executed correctly, data is retrieved and also updated correctly. More on database testing could be a load on DB, we will address this in web load or performance testing below.

In testing the functionality of the websites, the following should be tested:

Links
i. Internal Links
ii. External Links
iii. Mail Links
iv. Broken Links

Video Content / Details of website for further learning (if any):
https://testco.com/blog/web-application-testing-case-study/

Important Books/Journals for further learning including the page nos.:
Web Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 28

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 26.10.2021**

Topic of Lecture: Selecting and Installing Software Testing Tools

Introduction: Test Automation is being considered as the most effective way to enhance coverage, efficiency and effectiveness of any software application. It is redefining the way the engineers perform testing operations. It is important to recognize the relationship between a tool and a technique. A technique is a procedure for performing an operation; a tool is anything that serves as a means to get something done.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Client
- Server
- Tool
- Technique
- Network
- Testing Approaches

Detailed content of the Lecture:

Selecting and Installing Software Testing Tools
Technique-It is a procedure for performing an operation.
Tool-It is anything that serves as a means to get something done.

Tools Available for Testing Software

- Boundary Value Analysis(BVA)
- Cause-Effect Graphic
- Checklist
- Code Comparison
- Data Dictionary
- Flowchart
- Inspection
- Risk Matrix
- Test Data
- Test Script
- Use Cases
- Walk Through

Tool Selection Process

- **Step 1**: Identify the Requirement for Tools
- **Step 2** : Evaluate the Tools and Vendors
- **Step 3** : Estimate Cost and Benefit

Step 4 : Make the Final Decision

Testing Tools Evaluations

- Test Planning and Management
- Testing Product Integration
- Internet/Intranet Testing
- Ease of Use
- GUI and Client/Server Testing
- Load and Performance Testing
- Methodologies and Services

Video Content / Details of website for further learning (if any):
https://www.oreilly.com/library/view/effective-methods-for/9780764598371/ch04.html

Important Books/Journals for further learning including the page nos.:
William Perry, "Effective Methods of Software Testing", Third Edition, Wiley Publishing. **(Page No:135-157)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 29

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 27.10.2021**

Topic of Lecture: Software Test Automation

Introduction: A**utomation Testing or Test Automation** is a software testing technique that performs using special automated testing software tools to execute a test case suite. On the contrary, Manual Testing is performed by a human sitting in front of a computer carefully executing the test steps.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Client
- Server
- Tool
- Technique
- Network
- Testing Approaches

Detailed content of the Lecture:

Software Test Automation

- Test Automation is the use of software separate from the software being tested to control the execution of tests and the comparison of actual outcomes with predicted outcomes
- It make use of specialized tools to control the execution of tests and compare the actual results against the expected results

Major keys to the success of software test automation

- Automation saves time as software can execute Test Cases faster than human do
- Automated Tests can be more Reliable
- Automation Helps in Immediate Testing
- Automation can Protect an Organization Against Attrition of Test Engineers
- Test Automation Opens up Opportunities for Better Utilization of Global Resources
- Certain Types of Testing Cannot be Executed without Automation
- Automation means End-to-End, not Test Execution Alone

Essentials Needs of Software Test Automation

- A dedicated work force for test automation
- The commitment from senior managers and engineers
- The dedicated budget and project schedule
- A well-defined plan and strategy
- Talent engineers and cost-effective testing tools
- Maintenance of automated software tests and tools.

Advantages of Automation Testing

- Automation vastly increases the test coverage
- Automation optimizes the testing speed
- Automation improves the testing quality
- Automation reduces the cost of testing
- Automation reduces the test execution time

Disadvantages of Automation Testing

- Debugging the test script is a major issue. It may lead to consequences if any error is present in the test script
- Proficiency is required to write automation test scripts
- Test maintenance is costly
- Maintenance of test data is difficult.

Video Content / Details of website for further learning (if any):
https://www.journaldev.com/25157/automation-testing

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:387-390)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 30

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 02.11.2021**

Topic of Lecture: Skills needed for Automation

Introduction: Automation Testing is the key to the business success of the software industry. With automation, you can expand your business to a larger audience saving both time and effort. As an automation tester, it is very essential to have certain skills which help in testing the application better.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Client
- Server
- Tool
- Technique
- Network
- Testing Approaches

Detailed content of the Lecture:

Skills needed for Automation

- Automation testers should be well aware of the manual testing process and the test techniques that are adopted in the testing phase.
- It can be challenging and time-consuming for automation testers to design test scripts without having good functional knowledge of the application.
- If the automation test team has good exposure to the functionality, they can achieve good test coverage and better test accuracy.

Capture/Playback and Test Harness Tools (First Generation)

- Test harness tools are the category of capture/playback tool used for capturing and replaying sequence of tests.

Data Driven Tools (Second Generation)

- This method help in developing test scripts that generate the set of input conditions and corresponding expected output.

Action Driven(Third Generation)

- This technique enables a layman to create automated tests. No input and expected output conditions required for running the tests.

Software Test Automation Process

Step 1: Test Automation Planning
Step 2: Test Automation Design
Step 3: Test Tool Development
Step 4: Test Tool Deployment
Step 5: Review and Evaluation

Video Content / Details of website for further learning (if any):
https://www.softwaretestinggenius.com/what-skills-are-needed-for-automated-testing/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:392-293)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 31

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 09.11.2021**

Topic of Lecture: Scope of Automation

Introduction: Automation is an effective means of eliminating/reducing manual effort during regression and functional test case execution. Also the chances of defect escape will be reduced considerably, since human mistakes will not occur once the script is fully developed. Hence the automation is more economical considering the fact of time and quality.

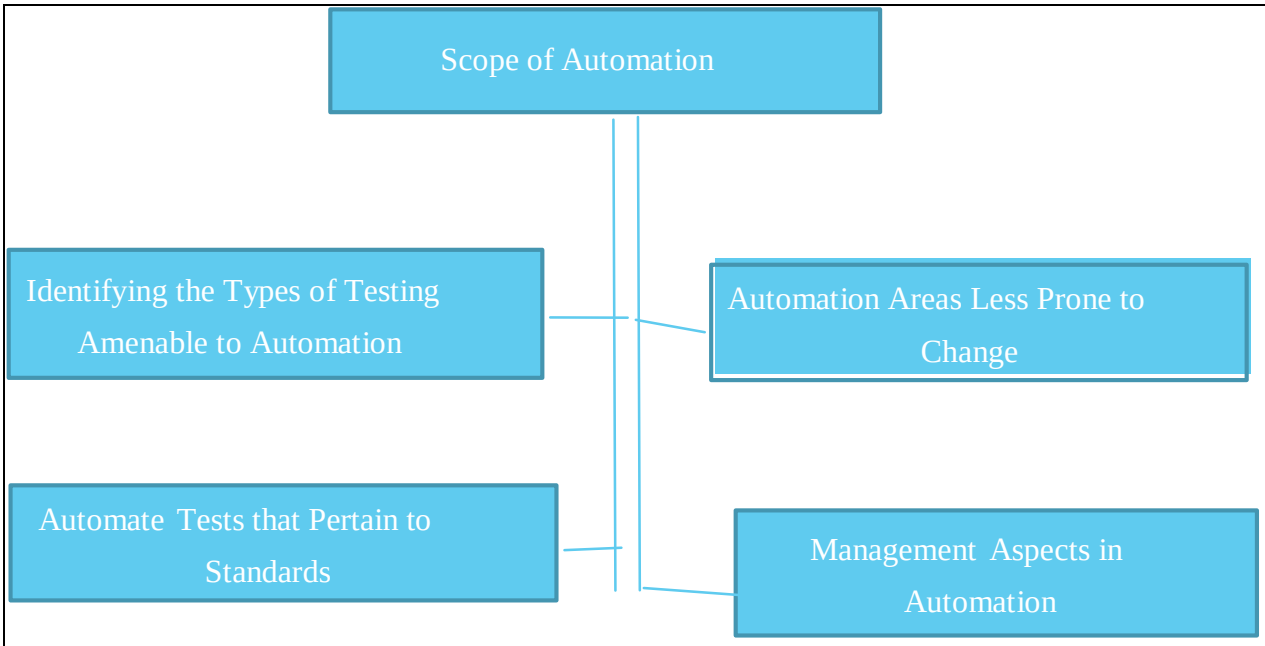
Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Client
- Server
- Tool
- Quality
- Network
- Testing Approaches

Detailed content of the Lecture:

Scope of Automation

- Identifying the Types of Testing Amenable to Automation
- Automation Areas Less Prone to Change
- Automate Tests that Pertain to Standards
- Management Aspects in Automation



The Manual testing is considered as the preliminary testing phase which generally evaluates the behavior of the app developed, by performing the step by step assessment of the requirement specification analysis record. The prime objective of manual testing is to ensure that the app works extraordinarily and fine without any sort of bugs and functional defects and also as per the requirement specification documents.

- To augment their skills and be more aware of the programming critical concepts and how the development works.
- Aware of how the code works.
- Learn more about how the business functions to improve ROI.
- Capacity to indulge in management activities.

Video Content / Details of website for further learning (if any):
https://shormistha4.medium.com/what-is-the-scope-of-automation-testing-or-manual-testing-in-the-future-642a6ccc3fdb

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:394-395)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 32

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 10.11.2021**

Topic of Lecture: Design and Architecture for Automation

Introduction: Architecture is the process of planning, designing and constructing a building or any other structures. For an automation testing project, architecture is as important as it is for constructing a building. The best architecture design is one that won't cost much but will continue to drive the economic flow. Designing and building architecture takes time, money, and collaboration.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tool

Detailed content of the Lecture:

Design and Architecture for Automation

- Design and Architecture is an important aspect for Automation
- Architecture for test automation involves two major heads- a test infrastructure that covers a test case database and a defect database/defect repository.

A framework is set of automation guidelines which help in

- Maintaining consistency of testing
- Improve test structuring
- Minimum usage of code
- Less maintenance of code
- Improve re-usability
- The training period of using the tool can be reduced

Four types of framework used in automation software testing

- Data Driven Automation Framework.
- Keyword Driven Automation Framework.
- Modular Automation Framework.
- Hybrid Automation Framework

Components of Test Automation

1. **External Modules**

- Test case Database
- Defect Repository

2. **Internal Modules**

- Scenario Module
- Configuration File Module
- Test Case Module
- Test Framework Module
- Tools Module
- Result Module
- Report Generator Module
- Report/Metric Module

Advantages of Software Test Automation

- Saves time
- More reliable
- Helps in immediate testing
- Security
- It opens up opportunities for better utilization of global resources
- It can free the test engineers from mundane task

Video Content / Details of website for further learning (if any):
https://medium.com/@assaini007/automation-architecture-design-pattern-5f81817ed530

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:396-399)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 33

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 12.11.2021**

Topic of Lecture: Requirements for a Test Tool

Introduction: Test automation tool selection is one of the most important steps before starting automation in any organization. It is important because the tool will greatly affect your whole automation effort. If the tool is good and gives you required features, the automation becomes easier and effective.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

Requirements for a Test Tool

- Automation testing success largely depends on the selection of right testing tools.
- It takes a lot of time to evaluate relevant automation tools available in the market.
- But this is a must one-time exercise that will benefit your project in long run.

- Test case/suite expandability
- Reuse of code for different types of testing
- Automation setup and clean up
- Independent of test cases
- Remote execution of test cases
- Portability of different platforms
- Requirements Considerations
- Software Under Test

- Tools and environment
- Test management
- Organizational situation
- Automation architecture

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/automation-testing-tutorial-4/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:402-406)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 34

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 13.11.2021**

Topic of Lecture: Challenges in Automation

Introduction: Automated testing Software is the methodology that helps to validate the functioning of the software before it is moved to production. In this process, automated testing tools are used by the QA teams for executing the test scripts.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

Challenges in Automation

- Test automation is an essential subset of software testing. By using automated testing, we can expedite the process of software validation and increase testing coverage.
- A lot of challenges in applying test automation for applications under test (AUT).
- Without overcoming these challenges, testers might face countless nightmares that can cause software automated testing failure.
- This topic aims to outline the top five challenges that have the highest impact on the overall automation test effort and project success.
- With the use of automated software testing tools, QA teams can quickly test the software, prepare the defect reports, and compare the software results with the expected results.

- Also, this automated testing process provides several benefits such as faster delivery, eases regression testing time and also ensures quality software along with reducing manual testing efforts.
- Thus, the QA teams can function faster and help to push the software to production quickly as per the given timelines as automated testing ensures faster and quality releases leveraging automated testing tools.
- Management commitment is an important challenge of test automation.
- It takes time/effort and pays off in the long run.
- It needs initial outlay of money as well as a steep learning curve for test engineers before it can start paying off.
- The main challenge is because of the heavy front loading of costs of test automation and managements starts to look for an early payback.

Video Content / Details of website for further learning (if any):
https://www.katalon.com/resources-center/blog/top-challenges-in-test-automation/

Important Books/Journals for further learning including the page nos.:
Srinivasan Desikan and Gopalaswamy Ramesh, "Software Testing – Principles and Practices", Pearson Education, 2009**(Page No:416-417)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 35

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 19.10.2021**

Topic of Lecture: Tracking the Bug , Debugging

Introduction: Debugging is the process of fixing a bug in the software. In other words, it refers to identifying, analyzing and removing errors. This activity begins after the software fails to execute properly and concludes by solving the problem and successfully testing the software. It is considered to be an extremely complex and tedious task because errors need to be resolved at all stages of debugging.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

Tracking the Bug

- Bug tracking is the process of capturing, reporting, managing data on bugs that occur in a software.
- Bug tracking is a process used by quality assurance personnel and programmers to keep track of software problems and resolutions.
- The goal is to maintain high product quality, using two types of services:
  1. Task Management Tools
  2. Bug Capturing Tools

Debugging

- Debugging is the process of analyzing and locating bugs when software does not behave as expected.
- Debugging is a diagnostic process, which diagnoses the bugs in the software product that enables it not to behave as expected.
- When a test case uncovers an error, debugging is the process that results in the removal of the error.

Debugging Process

- During debugging, errors are encountered that range from less damaging (like input of an incorrect function) to catastrophic (like system failure, which lead to economic or physical damage).
- Once errors are identified in a software system, to debug the problem, a number of steps are followed.

1) Defect Confirmation/Identification: A problem is identified in a system and a defect report is created.

2) Defect Analysis: If the defect is genuine, the next step is to understand the root cause of the problem.

3)**Defect Resolution**: Once the root cause of a problem is identified, the error can be resolved by making an appropriate change to the system by fixing the problem

Debugging Strategies

Debugging is locating the problem source by binary partitioning through working hypotheses that predict new values to be examined.

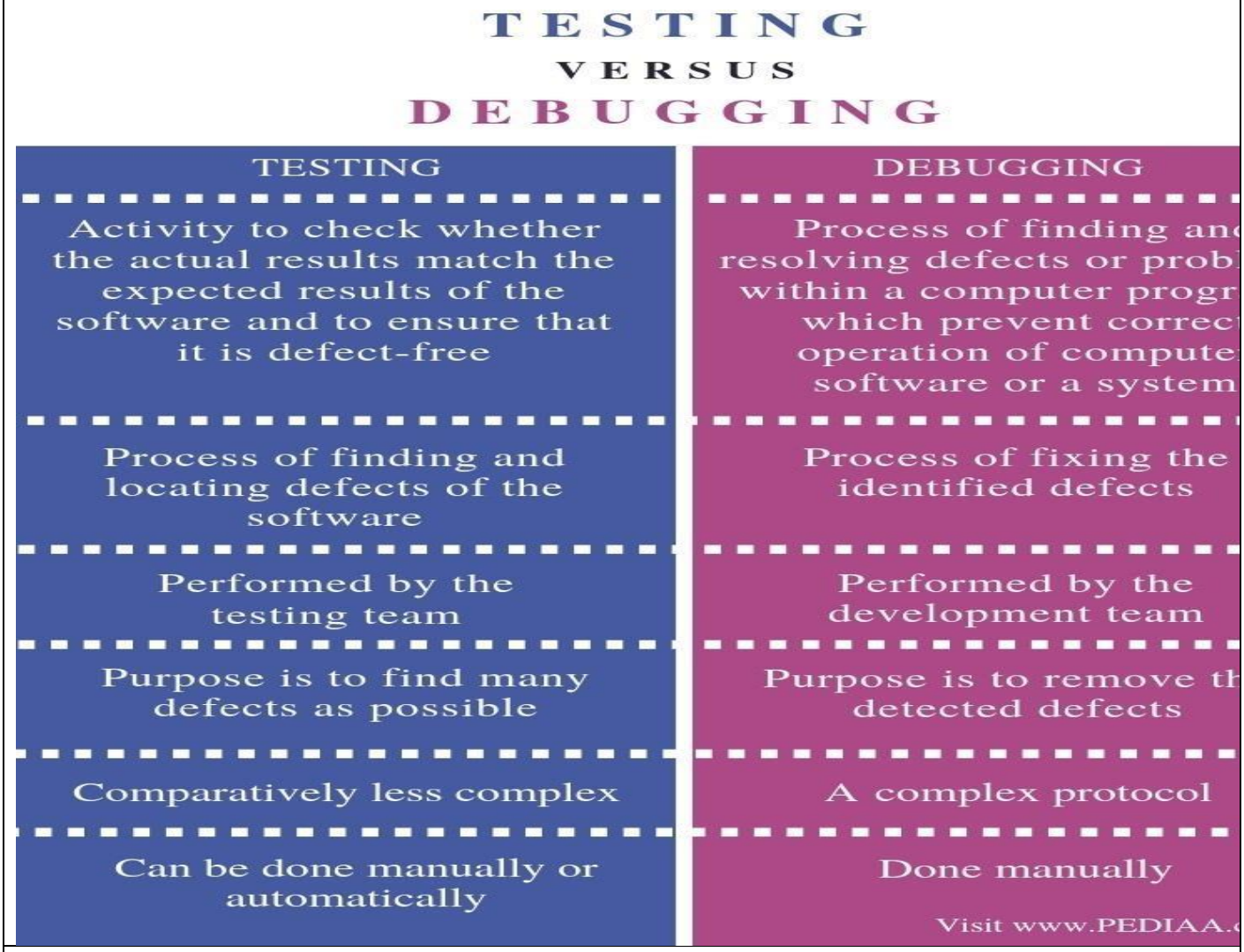
Broadly there are three categories of debugging approaches:

1) **Brute Force:** This method is most common and least efficient for isolating the cause of a software error. We apply this method when all else fail. In this method, a printout of all registers and relevant memory locations is obtained and studied. All dumps should be well documented and retained for possible use on subsequent problems.

2) **Back Tracking Method:** It is a quite popular approach of debugging which is used effectively in case of small applications. The process starts from the site where a particular symptom gets detected, from there on backward tracing is done across the entire source code till we are able to lay our hands on the site being the cause. Unfortunately, as the number of source lines increases, the number of potential backward paths may become unmanageably large.

3) **Cause Elimination:** The third approach to debugging, cause elimination, is manifested by induction or deduction and introduces the concept of binary partitioning. This approach is also called induction and deduction. Data related to the error occurrence are organized to isolate potential causes. A “cause hypothesis” is devised and the data are used to prove or disprove the hypothesis. Alternatively, a list of all possible causes is developed and tests are conducted to eliminate each. If initial tests indicate that a particular cause hypothesis shows promise, the data are refined in an attempt to isolate the bug.

Difference between Debugging and Testing



Video Content / Details of website for further learning (if any):
https://www.geeksforgeeks.org/software-engineering-debugging/

Important Books/Journals for further learning including the page nos.:
Naresh Chauhan , "Software Testing Principles and Practices " Oxford University Press , New Delhi ,2010**(Page No:503-510)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 36

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : IV – Test Automation **Date of Lecture: 23.11.2021**

Topic of Lecture: Case study using Bug Tracking Tool.

Introduction: Bug tracking is the process of capturing bugs in a software and managing the data of all bugs till date. The aim of any bug tracking is to maintain high product quality and report any issue which might lead to process discrepancy at any stage.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:
Case study using Bug Tracking Tool

- Bug tracking tools can be useful to execute test cases.
- Defect management tools are important to enhance the software quality while saving time and money.

Traceability to Failed Tests

- An integrated bug tracker system is important to trace specific failed test cases, so that required actions can be taken to make fixes accordingly.

Enhanced Collaboration & Team Work

- Bug tracking tools play a critical role in enhancing communication between the team members working towards fixing bugs and discovering applications.

Comprehending Defect Trends

- A bug tracking system helps engineers keep track of issues in the past, and create a bug-free environment with high-quality products.

Enhanced Customer Satisfaction

- Bug tracking tools are critical to resolve issues in your product or website at the early stage and deliver enhanced service to customers.

Bugzilla

- Bugzilla is one of the best web-based bug tracking tools, which is used by project managers for the bug and issue management.
- It also provides an advanced dashboard for keeping all your automated charts and reports in an organized manner.
- Bugzilla helps with duplicate bug detection and time tracking and allows mentioning notes for a bug so that the team can work on it.

Features

- Custom prefixes and filters
- Complex search algorithm
- Custom bug list formats
- Duplicate detection system
- High security layers to keep bugs away

Cost to Upgrade: It's an open source bug tracker tool, and all its features are available for free.

Redmine

- Redmine open source bug tracking tool is compatible with MySQL, PostgreSQL, Microsoft SQL, and SQLite.
- It supports more than 34 languages to help software developers and project managers prevent bugs from affecting the product quality.

Features

- It helps manage multiple projects simultaneously
- Controlled access based on user roles
- Ensures agile project management with calendar and Gantt chart reporting
- State of the art issue tracking solution

Cost to Upgrade: This is an open source bug tracking tool with free access.

Video Content / Details of website for further learning (if any):
https://www.softwaretestinghelp.com/popular-bug-tracking-software/

Important Books/Journals for further learning including the page nos.:
Net Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 37

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V- Software Testing and Quality Metrics Date of Lecture: 24.11.2021

Topic of Lecture: Six-Sigma , TQM

Introduction: Six Sigma is a methodology focused on creating breakthrough improvements by managing variation and reducing defects in processes across the enterprise.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Tool
- Technique
- Probability
- Statistics
- Quality Control
- Knowledge Management
- Process Management

Detailed content of the Lecture:

Six-Sigma

- Six-sigma is a highly disciplined process that helps us focus on developing and delivering near-perfect product and services.
- It is a measure of **quality** that strives for near **perfection.**
- Six-Sigma is a **disciplined, data-driven approach and methodology** for **eliminating defects** in any process- from production to transactional and from product to service.

Features of Six Sigma

- Improving Processes.
- Lowering Defects.
- Reducing process variability.
- Reducing costs.
- Increasing customer satisfaction.
- Increased profits.

Key Concepts of Six Sigma

- **Critical to Quality** – Attributes most important to the customer
- **Defect**-Failing to deliver what the customer wants.
- **Process Capability**-What your process can deliver
- **Variation**-What the customer sees and feels
- **Stable Operations**-Ensuring consistent, predictable process to improve what the customer sees and feels
- **Design for Six Sigma**-Design to meet customer needs and process capability

Six Sigma Life Cycle(SLC)

- **Define**-Project goal and customer deliverables.
- **Measure**-To determine the current performance in respect to the defined values.
- **Analyze**-Compare the output values with the defined ones and finding out the root cause of the defects.
- **Improve**-Improvement in the process in such way to get an output with less deviation & aimed at the elimination of defects.
- **Control**- It involves controlling the future process performance in such a way to get the output.

Benefits of Six-Sigma

- Generates sustained success.
- Sets a performance goal for everyone.
- Enhances value to customers.
- Accelerates the rate of improvement.
- Promotes learning and cross-pollination.
- Executes strategic change.
-

TQM-Total Quality Management

TQM is the foundation for activities, which include:

- Commitment by senior management and all employees.
- Meeting customer requirements.
- Improvement teams.
- Reducing service costs.
- Employee involvement and empowerment.
- Reducing development cycle times.
- Focus on improvement plans.

TQM principles can be grouped in to the following practical concepts

- Customer focus
- Leadership.
- Teamwork.
- Continuous improvement process.
- Measurement.
- Benchmarking as a driver to improvement in a competitive environment.

Key Elements of TQM

TQM aims to **enhance the quality** of products and services.

The **Key elements of TQM** are

- Processes
- People
- Management system
- Performance measurement

Requirements of TQM

TQM concentrates on enhancing quality, it is important an organization constantly attempt to integrate quality in products.

Requirements are

- Commitment.
- Communication.
- Culture.

Video Content / Details of website for further learning (if any):
http://discover6sigma.org/post/2005/10/introduction-to-six-sigma/

Important Books/Journals for further learning including the page nos.:
Dale H. Besterfiled , "Total Quality Management", Pearson Education Asia, Third Edition, Indian Reprint (2011)**(Page No:147-149,Page No:5-14)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 38

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 26.11.2021

Topic of Lecture: Complexity Metrics and Models

Introduction: Software Complexity metrics is a standard of measure that contains many activities which involve some degree of measurement. It can be classified into three categories: product metrics, process metrics, and project metrics. Reliability models are developed and studied by researchers and software reliability practitioners with sophisticated skills in mathematics and statistics, quality management models are developed by software quality professionals and product managers for practical project and quality management.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Metrics
- Models
- Software
- Design
- Quality
- Errors
- Tools

Detailed content of the Lecture:

Complexity Metrics and Models

- Complexity is a measure of the resources which must be expanded in developing, maintaining, or using a software product.
- Associated measure of complexity is the number of software errors.
- Cyclomatic complexity is one such complexity measure in which the complexity of a module is the number of independent cycles in the flow graph of the module.

Fundamental software attributes measure on complexity metrics are as follows

- Lines of Code
- Halstead Software Science
- Cyclomatic Complexity
- Syntactic Constructs
- Structure Metrics
- Hybrid Metrics.

Lines of Code

- A line of code is any line of program text that is not a comment/blank line, regardless of the number of statements/fragments of statements on the line.
- This specifically includes all lines containing program headers, declarations and executable and non-executable statements

Few metrics are

- Error per KLOC
- Defects per KLOC
- Page of documentation per KLOC
- Error per person-month
- LOC per person-month
- Comments and blank lines are not counted as instructions.

Halstead Software Science

- **Halstead complexity metrics** were developed by the Maurice Halstead as a means of determining a **quantitative measure of complexity** directly from the **operators** and **operands** in the module to measure a program modules complexity directly from source code.

Number of operators and operands.

- Number of unique operators(n1)
- Number of unique operands(n2)
- Total number of operators(N1) and
- Total number of operands(N2)

Cyclomatic Complexity

Cyclomatic Complexity is one such complexity measure in which the complexity of a module is the number of independent cycles in the flow graph of the module.

3 different ways to compute the cyclomatic complexity:

Method 1: Given a control flow graph G of a program, the cyclomatic complexity V(G) can be computed as: **V(G)=E-N+2.**

Where N= number of nodes of the control flow graph

E=number of edges in the control flow graph

Method 2: An alternative way of computing the cyclomatic complexity of a program from an inspection of its control flow graph is as:

V(G)=Total number of bounded areas +1

In program control flow graph G, any region enclosed by nodes and edges can be called as a bounded area

Method 3: It can be easily computed by computing the number of decision statements of the program. If N is the number of decision statement of a program, then the McCabe's metric is equal to N+1

Syntactic Constructs

- **Binder and poore** empirically supported the concept of **local software quality metrics** whose formulation is based on **software syntactic attributes.**
- In McCabe's cyclomatic complexity index is summary index of **binary decisions.** It does not distinguish different kinds of control flow complexity such as **loops** versus **IT-THEN-ELSES**.

Structure Metrics

- Structure metrics **try to take into account the interactions between** modules in a product **and** quantify **such interactions.**

- **Most common design structure metrics are the fan-in and fan-out** metrics(yourdon and constantine and myers**).**

Cp =(Fan-In*Fan-Out)²

Fan-in is the number of local flows into a procedure plus the global data structures from which a procedure retrieves information.

Fan-out is the number of local flows from a procedure plus the global data structures that the procedure updates.

Hybrid Metrics

- **Hybrid metrics** usually combine a structural complexity metric with a code complexity metric to provide a complete picture of the module's complexity.

- $Cp=Cip*(Fan\text{-}In*Fan\text{-}Out)^2$

Cip is any **code complexity**

Video Content / Details of website for further learning (if any):
https://flylib.com/books/en/1.428.1/complexity_metrics_and_models.html

Important Books/Journals for further learning including the page nos.:
Net Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 39

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 01.12.2021

Topic of Lecture: Quality Management Metrics

Introduction: Software quality metrics are quantifiable measures that could be used to measure different characteristics of a software system

Prerequisite knowledge for Complete understanding and learning of Topic:

- Software
- Quality
- Product
- Process
- Quality
- Defect

Detailed content of the Lecture:

Quality Management Metrics

- A **software metric** is a standard of **measure of a degree** to which a **software system possesses** some property.

Software quality metrics can be classified into **three** categories:

- **Product Metrics**: Describes the characteristics of the product such as size, complexity, design features, performance and quality level.
- **Process Metrics**: It can be used to improve software development and maintenance.
- **Project Metrics**: It describe the project characteristics and execution.

Mean Time to Failure(MTTF)

- **Mean Time to Failure(MTTF)** is the **time between failures**.
- This metric is mostly used with **safety critical systems** such as the **airline traffic control systems, weapons**. It is commonly used to describe the **reliability of non-repairable system**.
- It describes the average time a collection of systems runs until the next system failure.

The Defect Density Metric

- It measures the defects relative to the software size expressed as **lines of code/function points i.e., it measures code quality per unit**.
- This metric is used in many **commercial software systems**.
- **Defect Density** is defined as the number of confirmed defects which have been detected by the software tester in the particular software during a defined period of the development.

Percentage Test cases Failed=(Deflective test cases/Total test cases)*100 Customer Problems Metric

Defect Density During Machine Testing

- **Defect rate** during formal machine testing is **correlated with the defect rate in the field**.
- The simple metric of defects per **KLOC/Function point** is a good indicator of **quality**, while the software is still being tested.

Defect Arrival Pattern During Machine Testing

- The overall defect density during testing will provide only the summary of the defects.It includes the defect arrivals/defects reported during the testing phase by time interval (eg.Week).

Phase-Based Defect Removal Pattern

- **Phase-based defect removal** uses the defect removal effectiveness (DRE) metric.
- This is simply the defects removed during each development phase divided by the defects latent in the product, times 100 to get the result as a percentage.

Defect Removal Effectiveness

- Defect removal efficiency is the percentage of defects removed from software.
- It is the ratio of defects removed to defect escapes and defects injected. Defect removal efficiency is an indicator of how good/effective the software appraisal process has been.

Metrics for Software Maintenance

Fix Backlog and Backlog Management Index

- Fix backlog is a workload statement for software maintenance. It is related to both the rate of detects arrivals and the rate at which fixes for reported problems become available.

Fix Response Time and Fix Responsiveness

- For many software development organizations, guidelines are established on the time limit within which the fixes should be available for the reported defects.

Fix Quality

- **Fix quality** or the number of defective fixes is another important **quality metric for the maintenance phase**.
- A fix is defective if it did not fix the reported problem/ if it fixed the original problem but injected a new defect.
- A defective fix can be recorded in two ways: Record it in the month it was discovered or record it in the month the fix was delivered.

Video Content / Details of website for further learning (if any):

https://www.tutorialspoint.com/software_quality_management/software_quality_management_metrics.htm

Important Books/Journals for further learning including the page nos.:

Dale H. Besterfiled , "Total Quality Management", Pearson Education Asia, Third Edition, Indian Reprint (2011)**(Page No: 253-291)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 40

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 03.12.2021

Topic of Lecture: Availability Metrics , Defect Removal Effectiveness

Introduction: In this Internet age of network computing, one of the most critical quality attributes is system and network availability, along with reliability and security. Requirements for high availability by mission-critical operations have existed since society became reliant on computer technologies.

Prerequisite knowledge for Complete understanding and learning of Topic:

- System
- Defect
- Metrics
- Product
- Quality
- Tools
- Security

Detailed content of the Lecture:

Availability Metrics

- Availability is the amount of time a system is working at its full functionality during the time it required to do so.
- The key metrics involved in measuring availability are Mean Time between Failure(MTBF), sometimes referred to as Mean Time To Failure(MTTF), and Mean Time To Repair(MTTR).

A=MTRF/(MTBF+MTTR)

Defect removal effectiveness

- This metric can be calculated for the entire development process, for the front-end before code integration and for each phase. It is called **early defect removal** when used for the front-end and **phase effectiveness** for specific phases.
- The higher the value of the metric, the more effective the development process and the fewer the defects passed to the next phase or to the field.
- This metric is a key concept of the defect removal model for software development.

- One of most significant challenges in calculating a DRE percentage is determining the total number of defects introduced to the system by the project, there are a number of ways how this can be determined –

- **Defect Seeding** –Defect Seeding has been used in a number of academic studies but is rarely used in real world testing where timeframes and budgets are often already stretched, creating representative defects is a difficult and time consuming activity.

  The total number of defects in a application can be extrapolated to = total defects found during testing x (total defects seeded/total seeded defects detected)

- **Defect Estimation** – This involves estimating the number of defects within a system based on previous deliverables and industry experience. This is a technique that would be unlikely give a truly accurate defect count, and will be of more value as an input into the initial Test Planning estimates.

- **Defect Counting** – This involves combining the "number of defects found during testing" with the "number of defects identified in Production traced back to the project". This method will always give a lower value than the true number of defects introduced by the project as –Not all defects manifest themselves as failures in Production Cosmetic and Usability related defects are often less likely to be raised in Production.

Defects are usually classified as being introduced in the following areas –

- Requirements
- Design
- Build
- Deployment to Production

For an iterative project it is also good practice to record the iteration in which the defects were introduced.

Video Content / Details of website for further learning (if any):
https://flylib.com/books/en/1.428.1/defect_removal_effectiveness.html

Important Books/Journals for further learning including the page nos.:
Net Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 41

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V (Software Testing and Quality Metrics Date of Lecture: 07.12.2021

Topic of Lecture: FMEA

Introduction: The FMEA process will now be utilized to determine the risks associated with the manner in which the product could potentially fail to meet the design requirements.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

FMEA

- **Failure Mode And Effects Analysis (FMEA)** is a risk management technique.
- FMEA is an important technique to develop **ISO 9000-Complaint quality systems** by both evaluating reliability and determining the effects of system and equipment failure.
- The **goal** of FMEA is to identify potential design and process failure early in a project, when they are relatively easy and inexpensive to correct.
- FMEA is used heavily in the **health-care devices** and the **analysis of health care and procedures**.

Steps used to identify the highest priority actions to be taken in FMEA

Step 1: Determine the severity of rating.

Step 2: Determine the occurrence rating.

Step 3: Determine the criticality.

Step 4: Determine the detection rating.

Step 5: Calculate the risk priority rating.

Types of FMEA

- **System FMEA-** Focuses on global system functions.
- **Design FMEA-** Focuses on component under subsystems.
- **Process FMEA-** Focuses on manufacturing and assembly processes.
- **Service FMEA**-Focuses on service function.
- **Software FMEA**-Focuses on software function.

Procedure of FMEA

- FMEA is a team approach, by members from all sections of the organization.
- It is a cross functional management, where team leader is responsible coordinating all activities, which include meeting time, place and frequency, corrective action and preventive actions.

Following are the steps to conduct FMEA:

Step 1: Product/Process and its function must be understood.

Step 2: Block diagram of Product/Process has to be created and developed.

Step 3: Header on FMEA form has to be completed.

Step 4: The functions are listed logically in the FMEA form.

Step 5: Then failure modes are identified.

Step 6: A failure mode is one component can cause failure in another component.

Step 7: The effects of each failure mode have to be described.

Step 8: The causes of each failure mode have to be identified.

Step 9: Probability factor indicating the frequency of occurrence is considered.

Step 10: Design and process have to identified before it reaches the customer.

Step 11: Assessment of detection rating has to be done.

Step 12: RPN=Severity X Probability X Detection.

Step 13: Recommended action have to be determined.

Step 14: Responsibility and target completion for these action have to be assigned.

Step 15: Revalidate each action by reassessing severity, probability and detection(RPN).

Limitations of FMEA

- Analysis of complex systems can be tedious and difficult.
- Failure effects cannot be analyzed.
- Successful completion requires expertise, experience and good team skills.
- Dealing with data redundancies can be difficult.
- Can be costly and time consuming.

Video Content / Details of website for further learning (if any):
https://asq.org/quality-resources/fmea

Important Books/Journals for further learning including the page nos.:
Dale H. Besterfiled , "Total Quality Management", Pearson Education Asia, Third Edition, Indian Reprint (2011)(**Page No:377-404)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

LECTURE HANDOUTS

L 42

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 08.12.2021

Topic of Lecture: Quality Function Deployment

Introduction: QFD is the process of translating the Voice of the Customer (needs, expectations, etc.,) into **design requirements**. Those design requirements are now the input to the Design FMEA

Prerequisite knowledge for Complete understanding and learning of Topic:

- Quality
- Design
- Server
- Design
- Function
- Product
- Tools

Detailed content of the Lecture:

Quality Function Development (QFD)

- **QFD** is a very systematic and organized approach of taking customer needs and demands in the consideration when designing a new products and service or when improving existing products and services.

QFD identifies three types of requirements:

1. Normal Requirements.
2. Expected Requirements.
3. Exciting Requirements.

In meeting with customer, functional deployment is used to determine the value of each function.

- Information deployment identifies both the data objects and events that the system must consume and produce.
- Task deployment examines the behavior of the system.
- QFD uses Customer Voice Table- that is reviewed with the customer.
- The Purpose of QFD
- The purpose of QFD is three fold.
- Firstly, it allows us to get higher quality products to market faster and at a lower cost.
- Secondly, we will achieve customer driven product design and, finally, it will provide a tracking system for future design or process improvements.

The results we can expect by carrying out the QFD studies are many:

• Better understanding of customer needs

• Improved organization on development projects

• Improved introduction to production

• Fewer design changes late in development

• Fewer manufacturing start-up problems

• Reputation for being serious about quality

• Increased business

Video Content / Details of website for further learning (if any):
https://elsmar.com/elsmarqualityforum/threads/fmea-qfd-quality-function-deployment-interlinked.31577/

Important Books/Journals for further learning including the page nos.:
Dale H. Besterfiled , "Total Quality Management", Pearson Education Asia, Third Edition, Indian Reprint (2011)**(Page No:315-347)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 43

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 13.12.2021

Topic of Lecture: Taguchi Quality Loss Function

Introduction: The quality loss function as defined by Taguchi is the loss imparted to the society by the product from the time the product is designed to the time it is shipped to the customer.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

Taguchi Quality Loss Function

- Genichi Taguchi has provided us with three concepts aimed at improving both product and process quality-quality robustness, quality loss function and target-oriented quality.
- This idea is to remove the effects of adverse conditions instead of removing the causes.
- A Quality Loss Function(QLF) identifies all costs connected with poor quality and shows how these costs increase as the product moves away from being exactly what the customer wants.
- This costs include not only customer dissatisfaction but also warranty and service costs, internal inspection, repair and scrap costs and costs that can best be described as costs to society.

$L=D^2C$

L=Loss of society, D^2= Square of the distance from the target value,

C=Cost of the deviation at the specification limit

- Taguchi aims for the target because products produced near the upper and lower acceptable specifications result in higher quality loss function.

- Genichi Taguchi is a Japanese quality expert, known for the Quality Loss Function and for methodologies to optimize quality at the design stage – ―*robust design*.

- Taguchi received formal recognition for his work including Deming Prizes and Awards.

- Genichi Taguchi considers quality loss all the way through to the customer, including cost of scrap, rework, downtime, warranty claims and ultimately reduced market share.

- The Quality Loss Function gives a financial value for customers' increasing dissatisfaction as the product performance goes below the desired target performance.

- Equally, it gives a financial value for increasing costs as product performance goes above the desired target performance.

- Determining the target performance is an educated guess, often based on customer surveys and feedback.

- The quality loss function allows financial decisions to be made at the design stage regarding the cost of achieving the target performance.

Video Content / Details of website for further learning (if any):
http://www.brainkart.com/article/Taguchi---s-Quality-Loss-Function_5287/

Important Books/Journals for further learning including the page nos.:
Dale H. Besterfiled , "Total Quality Management", Pearson Education Asia, Third Edition, Indian Reprint (2011)**(Page No:561-611)**

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

LECTURE HANDOUTS

L 44

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 14.12.2021

Topic of Lecture: Cost of Quality

Introduction: Cost of quality (COQ) is defined as a methodology that allows an organization to determine the extent to which its resources are used for activities that prevent poor quality, that appraise the quality of the organization's products or services, and that result from internal and external failures.

Prerequisite knowledge for Complete understanding and learning of Topic:

- Resources
- End User
- Fault
- Design
- Quality
- Cost

Detailed content of the Lecture:

Cost of Quality

- **Cost of Quality (COQ)** is a measure that **quantifies the cost of control** and **the cost of failure of control.**
- It is one of the most **established, effective measures of quantifying and calculating the business values of testing.**

To measure this, the project and its budgeted expenses must be classified in to these four categories.

- **Prevention Costs**: Eg. Quality assurance costs.
- **Appraisal Costs:** Eg. Quality control costs.
- **Internal Failure Costs**: Eg. Cost of rework(Internal defects).
- **External Failure Costs**: Eg. Cost of rework(External defects).

Cost of Quality(COQ)=Cost of Control + Cost of Failure of Control where Cost of Failure of Control= Internal Failure Cost + External Cost of Quality(COQ)=

Cost of Control + Cost of Failure of Control

(Prevention Cost + Appraisal Cost) (Internal Failure Cost + External Failure Cost)

- Total Quality Costs represent the difference between the actual (current) cost of a product or service and what the reduced cost would be if there were no failure of products, or defects in the manufacturing process.

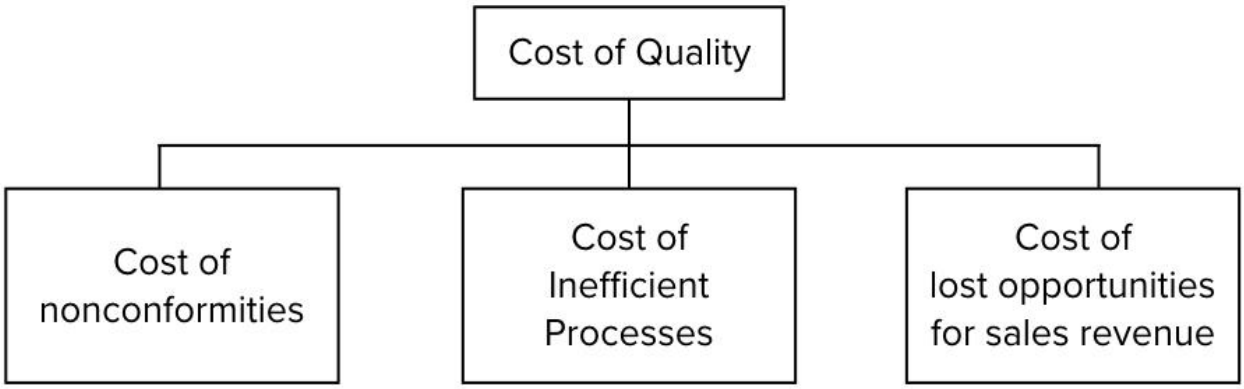


Figure : Components of the Cost of Quality

Video Content / Details of website for further learning (if any):
https://asq.org/quality-resources/cost-of-quality

Important Books/Journals for further learning including the page nos.:
Net Reference

Course Faculty

Verified by HOD

MUTHAYAMMAL ENGINEERING COLLEGE
(An Autonomous Institution)

(Approved by AICTE, New Delhi, Accredited by NAAC & Affiliated to Anna University)
Rasipuram - 637 408, Namakkal Dist., Tamil Nadu

Estd. 2000

IQAC

LECTURE HANDOUTS

L 45

MCA

II / III

Course Name with Code : 19CAC21/ Software Testing and Quality Assurance

Course Faculty : Mrs.M.Menakapriya

Unit : V - Software Testing and Quality Metrics Date of Lecture: 15.12.2021

Topic of Lecture: Case Study for Complexity and Object-Oriented Metrics.

Introduction: Object orientation has gained a wide adoption in the software development community. To this end, different metrics that can be utilized in measuring and improving the quality of object-oriented (OO) software have been proposed, by providing insight into the maintainability and reliability of the system. Some of these software metrics are based on cognitive weight and are referred to as cognitive complexity metrics

Prerequisite knowledge for Complete understanding and learning of Topic:

- Framework
- Client
- Server
- Design
- Quality
- Network
- Tools

Detailed content of the Lecture:

Case Study for Complexity and Object-Oriented Metrics.

- Object-oriented design and development is becoming very popular in today's software development environment.
- Object oriented development requires not only a different approach to design and implementation, it requires a different approach to software metrics.
- Since object-oriented technology uses objects and not algorithms as its fundamental building blocks, the approach to software metrics for object oriented programs must be different from the standard metrics set.
- Some metrics, such as lines of code and cyclomatic complexity, have become accepted as "standard" for traditional functional/ procedural programs, but for object oriented, there are many

proposed object-oriented metrics in the literature.

- Within this framework, nine metrics for object oriented are selected. These metrics include three traditional metrics adapted for an object-oriented environment, and six "new" metrics to evaluate the principle object-oriented structures and concepts.
- Programming complexity (or software complexity) is a term that includes many properties of a piece of software, all of which affect internal interactions.
- According to several commentators, there is a distinction between the terms complex and complicated.
- Complicated implies being difficult to understand but with time and effort, ultimately knowable.
- Complex, on the other hand, describes the interactions between a number of entities.
- As the number of entities increases, the number of interactions between them would increase exponentially, and it would get to a point where it would be impossible to know and understand all of them.

Context: Systems of systems (SoS) are highly complex and are integrated on multiple levels (unit, component, system, system of systems). Many of the characteristics of SoS (such as operational and managerial independence, integration of system into system of systems, SoS comprised of complex systems) make their development and testing challenging.

Contribution: This method provides an understanding of SoS testing in large-scale industry settings with respect to challenges and how to address them.

Method: The research method used is case study research. As data collection methods we used interviews, documentation, and fault slippage data.

Results: We identified challenges related to SoS with respect to fault slippage, test turn-around time, and test maintainability. We also classified the testing challenges to general testing challenges, challenges amplified by SoS, and challenges that are SoS specific.

Video Content / Details of website for further learning (if any):
https://wiki.aalto.fi/display/~mmantyla@aalto.fi/Testing+Highly+Complex+System+of+Systems+-+An+Industrial+Case+Study

Important Books/Journals for further learning including the page nos.:
Net Reference

Course Faculty

Verified by HOD