SNS COLLEGE OF ENGINEERING

Kurumbapalayam (Po), Coimbatore – 641 107

An Autonomous Institution

Accredited by NBA – AICTE and Accredited by NAAC – UGC with 'A' Grade

Approved by AICTE, New Delhi & Affiliated to Anna University, Chennai

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

COURSE NAME : 19CS511 SOFTWARE TESTING

III YEAR / V SEMESTER

Unit 2 - INTRODUCTION

Topic 1 : Evaluating test adequacy criteria

IT8076 SOFTWARE TESTING

Syllabus

UNIT I INTRODUCTION 9

Testing as an Engineering Activity – Testing as a Process – Testing Maturity Model- Testing axioms – Basic definitions – Software Testing Principles – The Tester‘s Role in a Software Development Organization – Origins of Defects – Cost of defects – Defect Classes – The Defect Repository and Test Design –Defect Examples- Developer/Tester Support of Developing a Defect Repository.

UNIT II TEST CASE DESIGN STRATEGIES 9

Test case Design Strategies – Using Black Box Approach to Test Case Design – Boundary Value Analysis – Equivalence Class Partitioning – State based testing – Cause-effect graphing – Compatibility testing – user documentation testing – domain testing - Random Testing – Requirements based testing – Using White Box Approach to Test design – Test Adequacy Criteria – static testing vs. structural testing – code functional testing – Coverage and Control Flow Graphs – Covering Code Logic – Paths – code complexity testing – Additional White box testing approaches- Evaluating Test Adequacy Criteria..

IT8076 SOFTWARE TESTING

UNIT III LEVELS OF TESTING 9

The need for Levels of Testing – Unit Test – Unit Test Planning – Designing the Unit Tests – The Test Harness – Running the Unit tests and Recording results – Integration tests – Designing Integration Tests – Integration Test Planning – Scenario testing – Defect bash elimination System Testing – Acceptance testing – Performance testing – Regression Testing – Internationalization testing – Ad-hoc testing – Alpha, Beta Tests – Testing OO systems – Usability and Accessibility testing – Configuration testing –Compatibility testing – Testing the documentation – Website testing

UNIT IV TEST MANAGEMENT 9

People and organizational issues in testing – Organization structures for testing teams – testing services – Test Planning – Test Plan Components – Test Plan Attachments – Locating Test Items – test management – test process – Reporting Test Results – Introducing the test specialist – Skills needed by a test specialist – Building a Testing Group- The Structure of Testing Group- .The Technical Training Program.

IT8076 SOFTWARE TESTING

UNIT V TEST AUTOMATION 9

Software test automation – skills needed for automation – scope of automation – design and architecture for automation – requirements for a test tool – challenges in automation – Test metrics and measurements – project, progress and productivity metrics.

Evaluating test adequacy criteria

Evaluating test adequacy criteria in software testing involves assessing the effectiveness and thoroughness of test cases in verifying the correctness and reliability of a software application. These criteria help determine whether a set of test cases is sufficient to validate the software under test (SUT) against its requirements. The following are some key test adequacy criteria, along with detailed explanations

Code Coverage Criteria

Statement Coverage: Ensures that each executable statement in the code has been executed at least once. It helps identify code that has never been executed.

Branch Coverage: Also known as decision coverage, this criteria ensures that each possible branch (true/false) of a decision point is executed. It helps in identifying untested logical paths.

Path Coverage: Involves executing all possible paths through the code. This is a comprehensive but often impractical criterion due to the potentially vast number of paths.

Condition Coverage: Ensures that each condition in a decision has been evaluated to both true and false.

Data Flow Coverage

This criteria focuses on the flow of data through the software, ensuring that the values of variables are adequately tested. Key aspects include:

Definition-Use (DU) Pairs: Testing that each variable's definition (where it gets a value) and its subsequent use (where it's accessed) are covered by the tests.

All-defs and All-uses: Ensures all definitions and uses of variables are tested, helping to identify data-related errors.

Specification-Based Criteria

These criteria derive test cases from the software's specifications, ensuring that all functional requirements are tested. Examples include:

Equivalence Partitioning: Divides input data into partitions where test cases are designed to cover each partition. The assumption is that if one case in a partition works correctly, all others will too.

Boundary Value Analysis: Focuses on testing the boundaries of input ranges, where errors are often found.

Cause-Effect Graphing: Involves creating a graphical representation of logical relationships between causes (inputs) and effects (outputs) to derive test cases.

Experience-Based Criteria

These criteria rely on the experience and intuition of testers. While less formal, they can be highly effective:

Exploratory Testing: Involves testers exploring the application without pre-defined test cases, using their experience to identify potential issues.

Error Guessing: Testers use their knowledge of common errors and past defects to create test cases that specifically target potential problem areas.

Evaluating Test Adequacy

To evaluate the adequacy of test criteria:

Measurement: Determine the percentage of the criteria met by the test cases (e.g., percentage of code coverage).

Assessment: Evaluate the uncovered areas to determine if they are critical and require additional testing.

Effectiveness: Analyze the detected defects against the criteria to assess the effectiveness of the testing process.

Improvement: Identify gaps in the test suite and refine the test cases or criteria to improve coverage and defect detection.

Challenges and Considerations

Completeness vs. Practicality: Some criteria, like path coverage, may be theoretically desirable but impractical due to the sheer number of test cases required.

Resource Constraints: The availability of time, budget, and testing resources can limit the extent to which criteria can be applied.

Risk-Based Approach: Focusing testing efforts on the most critical and risk-prone areas of the software can provide the best return on investment.

TEXT BOOKS:

1. Ricardo Baeza-Yates and Berthier Ribeiro-Neto, —Modern Information Retrieval: The Concepts and Technology behind Search, Second Edition, ACM Press Books, 2011.

2. Ricci, F, Rokach, L. Shapira, B.Kantor, —Recommender Systems Handbook‖, First Edition, 2011.

REFERENCES:

1. C. Manning, P. Raghavan, and H. Schütze, —Introduction to Information Retrieval, Cambridge University Press, 2008.

2. Stefan Buettcher, Charles L. A. Clarke and Gordon V. Cormack, —Information Retrieval: Implementing and Evaluating Search Engines, The MIT Press, 2010.


SNS COLLEGE OF ENGINEERING
COIMBATORE -107


sns
INSTITUTIONS

THANK YOU