Software Testing

Definitions of “TESTING”

- Hetzel: Any activity aimed at evaluating an attribute or capability of a program or system. It is the measurement of software quality.
- Beizer: The act of executing tests. Tests are designed and then executed to demonstrate the correspondence between an element and its specification.

Definitions of “TESTING” (cont’d)

- Myers: The process of executing a program with **the *intent* of finding errors.**
- IEEE: The process of exercising or evaluating a system or system component by manual or automated means to verify that it satisfies specified requirements or to identify differences between expected and actual results.

Fisherman’s Dilemma

- You have 3 days for fishing and 2 lakes to choose from. Day 1 at lake X nets 8 fish. Day 2 at lake Y nets 32 fish. Which lake do you return to for day 3?
- Does your answer depend on any assumptions?

Di Lemma

- In general, the probability of the existence of more errors in a section of a program is directly related to the number of errors already found in that section.

SDLC Phases

- Planning
  - Test Cases Plan
- After Requirements before Testing
  - Test Cases Design/Writeup
- Testing Phase
  - TC Execute

Stakeholders / Participants

- PM
  - Test Planning on the basis of SQE advice
- Programmer
  - Only WB
- SQ Persons
  - BB
- User
  - UATs

WB & BB

BlackBox Testing

- Functional Testing
- BB
    - Requirements
    - Boundary Value / Equivalence Class / Cause Effect
- Documents
    - Test Protocol
    - TTM (Test Traceability)
    - Test Forms
        - Test script

Start BB Testing

- From Requirements

- Relationship
  - 1R….1TC
  - 0R….TC (X)
  - 1R…Many TC

Testing Techniques

- **Black-Box:** Testing based solely on analysis of requirements (unit/component specification, user documentation, etc.). Also know as *functional testing.*
- **White-Box:** Testing based on analysis of internal logic (design, code, etc.). (But *expected* results still come from requirements.) Also known as *structural testing.*

Levels or Phases of Testing

- **Unit:** testing of the smallest programmer work assignments that can reasonably be planned and tracked (e.g., function, procedure, module, object class, etc.)
- **Component:** testing a collection of units that make up a component (e.g., program, package, task, interacting object classes, etc.)

Levels or Phases of Testing (cont’d)

- **Product:** testing a collection of components that make up a product (e.g., subsystem, application, etc.)
- **System:** testing a collection of products that make up a deliverable system

Other Types of Testing

- Integration: testing which takes place as sub-elements are combined (i.e., *integrated*) to form higher-level elements
- Regression: **re-**testing to detect problems caused by the adverse effects of program change
- Acceptance: formal testing conducted to enable the customer to determine whether or not to accept the system (acceptance criteria may be defined in a contract)

Other Types of Testing (cont’d)

- Alpha: actual end-user testing performed within the development environment
- Beta: end-user testing performed within the user environment prior to general release
- System Test Acceptance: testing conducted to ensure that a system is “ready” for the system-level test phase

Other Types of Testing (cont’d)

- Soak: testing a system version over a significant period of time to discover latent errors or performance problems (due to memory leaks, buffer/file overflow, etc.)
- Smoke (build verification): the first test after a software build to detect catastrophic failure (Term comes from hardware testing…)
- Lights out: testing conducted without human intervention – e.g., after normal working hours

Plan-Based Testing Process Activities

Test Planning

Test Design

Test Implementation

Test Execution

Execution Analysis

Result Documentation

Final Reporting

Exhaustive Testing is Exhausting

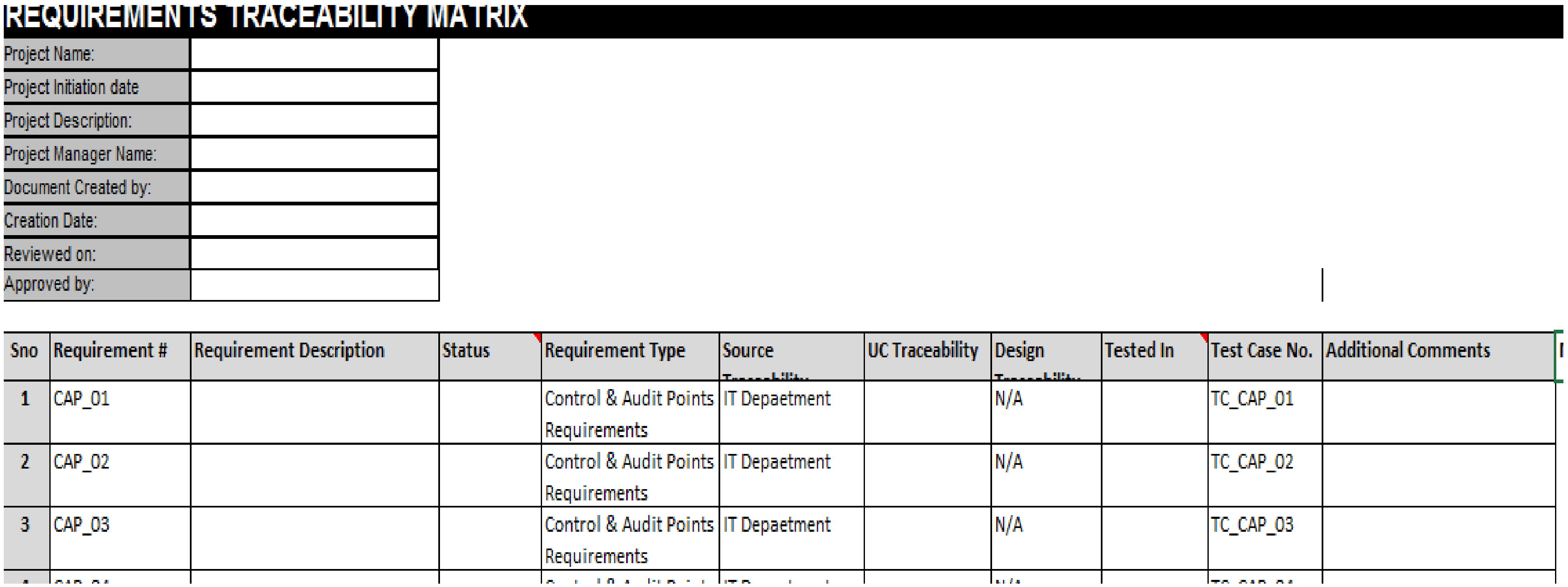
- **Situation:**
  - A module has 2 input parameters.
  - Word size is 32 bits.
  - Testing is completely automated: 100 nanoseconds are required for each test case.
- **Question:** How long would it take to test this module *exhaustively*, i.e., covering every possible combination of input values?

Vehicles for Continuous Process Improvement

- **Post-Test Analysis:** reviewing the results of a testing activity with the intent to improve its effectiveness
- **Causal Analysis:** identifying the causes of errors and approaches to eliminate future occurrences
- **Benchmarking:** general practice of recording and comparing indices of performance, quality, cost, etc., to help identify “best practices”

Test Case No.		**Test Status**		**Tester Name**	
Req. Reference No.		**Testing Date**		**Tester Signature**	

Purpose & Scope	
Test strategy:	**Testing Methodology:**
Test Script & Results	
Test Script:	
Expected Result:	Actual Result:
Exception & Corrective Action	
Comments & Conclusion	



REQUIREMENTS TRACEABILITY MATRIX

Project Name:	
Project Initiation date	
Project Description:	
Project Manager Name:	
Document Created by:	
Creation Date:	
Reviewed on:	
Approved by:	

Sno	Requirement #	Requirement Description	Status	Requirement Type	Source	UC Traceability	Design	Tested In	Test Case No.	Additional Comments
1	CAP_01			Control & Audit Points Requirements	IT Depaetment		N/A		TC_CAP_01	
2	CAP_02			Control & Audit Points Requirements	IT Depaetment		N/A		TC_CAP_02	
3	CAP_03			Control & Audit Points Requirements	IT Depaetment		N/A		TC_CAP_03	

Invalid and Unexpected Inputs

- Test cases must be written for INVALID and UNEXPECTED, as well as valid and expected, input conditions.
- In many systems, MOST of the code is concerned with input error checking and handling.

Anatomy of a Test Case

- What are the *parts* of a test case?
  1. a description of input condition(s)
  2. a description of expected results
- Where do ‘‘expected results’’ come from?

Black-Box Testing
Techniques I

Definition of Black-Box Testing

- Testing based solely on **analysis of requirements** (specification, user documentation, etc.).
- Also know as *functional* testing.
- Black-box techniques apply to *all* levels of testing (e.g., unit, component, product, and system).