TEST CASE DESIGN EFFECTIVENESS

The objectives of the test case design effectiveness metric is to,

(i) measure the "defect revealing ability" of the test suite and

(ii) use the metric to improve thetest design process.

During system-level testing, defects are revealed due to the execution of planned test cases. In addition to these defects, new defects are found during testing for which no test cases had been planned. For these new defects, new test cases are designed, which are called test case escaped (TCE). Test escapes occur because of deficiencies in the test design process. This happens because the test engineers get new ideas while executing the planned test cases. A metric commonly used in the industry to measure test case design effectiveness is the test case design yield (TCDY), defined as

$$TCDY = \frac{NPT}{NPT + \text{number of TCE}} \times 100\%$$

The TCDY is also used to measure the effectiveness of a particular testing phase. For example, the system integration manager may want to know the TCDY value for his or her system integration testing.

MODEL DRIVEN TEST DESIGN

MDTD is built on the idea that designers will become more effective and efficient if they can raise the level of abstraction. This approach breaks down the testing into a series of small tasks that simplify test generation. Then test designers isolate their tasks and work at a higher level of abstraction by using mathematical engineering structures to design test values independently of the details of the software or design artifacts, test automation, and Test Execution.

Different phases in MDTD:

MDTD can be done in 4 different phases. Each type of activity requires different skills, background knowledge, education, and training. It is better to use different sets of people depend on the situation.

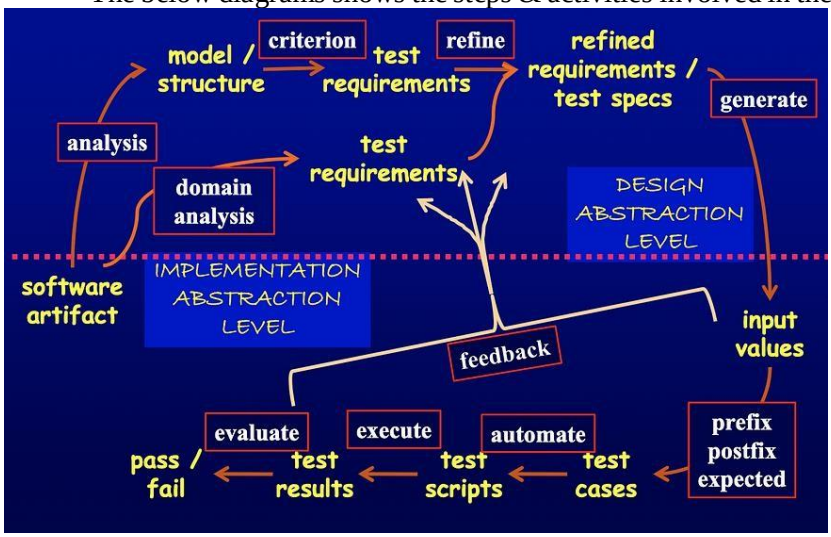
Test Design — This can be done in either Criteria-Based where Design test values satisfy coverage criteria or other engineering goals or in Human-Based where Design test values based on domain knowledge of the program and human knowledge of testing which is comparatively harder. This the most technical part of the MDTD process better to use experienced developers in this phase.

Test Automation — This involves embedding test values to scripts. Test cases are defined based on the test requirements. Test values are chosen such that we can cover a larger part of the application with fewer test cases. We don't need that much domain knowledge in this phase, however, we need to use technically skilled people.

Test Execution — The test engineer will run tests and records the results in this activity. Unlike the previous activities, test execution not required a high skill set such as technical knowledge, logical thinking, or domain knowledge. Since we consider this phase comparatively low risk, we can assign junior intern engineers to execute the process. But we should focus on monitoring, log collecting activities based on automation tools.

Test Evaluation — The process of evaluating the results and reporting to developers. This phase is comparatively harder and we expected to have knowledge in the domain, testing, and User interfaces, and psychology.

The below diagrams shows the steps & activities involved in the MDTD,



TEST PROCEDURES:

The Software Test Procedures describe the test preparations, test configuration, test cases, and test methods to be used to perform qualification testing of a computer software configuration item (CSCI) or a software system or subsystem. The test procedures also describe the expected test results and include bidirectional traceability to the requirements or a reference to the document containing that trace.

The following documents are useful when developing test procedures:

- Software Requirements Specification (SRS).
- Software Data Dictionary.
- Software Design Description.
- Interface Design Description.
- SW Change Requests_Problem Reports.

When writing test procedures, remember to:

- Include non-functional requirements, including safety, security, performance, etc.
- Ensure all requirements are covered by the full set of test procedures.
- Maintain the bidirectional test-to-requirements trace when modifying test procedures.
- Include test preparations for both software and hardware:
- Noting in the test procedure any dependencies in the order the test procedures must be run.
- Noting or setting the state of the system to that required to run the test procedure.
- Noting or setting the status of data values required to run the test procedure.
- Include tests to:
    - Confirm the software does what it is supposed to do.
    - Confirm the software does not do what it should not do.
    - Confirm the software behaves in an expected manner under adverse or off-nominal conditions.
    - Cover the range of allowable inputs, boundary conditions, false or invalid inputs, load tests, stress tests, interrupt execution and processing, etc.
    - Include performance testing.

If reusing test procedures, be sure to:

- Check that those procedures adhere to the content and helpful practice guidance above.
- Revise those test procedures to align with testing planned for the current project.