Chapter 6
Testing Tools and Measurements

Manual Testing :

- In Manual Testing , Testers **manually execute test cases** without using any automation tools.
- It requires a tester to play the role of an end user.
- Any new application **must be manually tested before** its testing can be automated.
- Manual testing requires **more effort**, but is necessary to check automation feasibility.
- Manual Testing does not require knowledge of any testing tool.
- One of the Software Testing Fundamental is **"100% Automation is not possible"**. This makes Manual Testing imperative.

Advantages of manual testing:

1. It is preferable for products with **short life cycles**.
2. It is preferable for products that have **GUIs** that **constantly change**
3. It requires **less time and expense to begin** productive manual testing.
4. Automation **can not replace human intuition, inference, and inductive reasoning**
5. Automation **can not change course in the middle** of a test run to examine something that had not been previously considered.
6. Automation tests are **more easily fooled** than human testers.

Disadvantages of manual testing:

1. Requires **more time or more resources**, some times both
2. **Performance testing** is **impractical** in manual testing.
3. **Less Accuracy**
4. Executing same tests again and again **time taking process** as well as **Tedious**.
5. **Not** Suitable for **Large scale projects** and **time bounded** projects.

Continued…

6. Batch Testing is not possible, for each and every test execution **Human user interaction is mandatory**.
7. Manual Test Case **scope is very limited.**
8. **Comparing large** amount of data is impractical.
9. Checking **relevance of search** of operation is **difficult**
10. **Processing change requests** during software maintenance **takes more time**.

Comparison between Automation Testing and Manual testing

Automation Testing	Manual Testing
Perform the **same operation** each time	Test execution is **not accurate all the time.** Hence not reliable.
Useful to execute the set of test cases **frequently.**	Useful when the test case only needs to run **once or twice**.
Fewer testers required to execute the test cases.	**Large number** of tester required
Platform **independent.**	It is Platform **dependent.**
Testers can **fetch complicated information** from code.	Does **not involve** in programming task to **fetch hidden information**.
Faster	**Slow**
Not Helpful in UI	Helpful in UI
High Cost	**Less** than automation.

Why Automated Testing?

- All work flows, all fields , all negative scenarios
- Difficult to test for multi lingual sites manually
- Does not require Human intervention
- Increases speed of test execution
- Increase Test Coverage
- Manual Testing can become boring and hence error prone

Which Test Cases to Automate?

Test cases to be automated can be selected using the following criterion to increase the automation ROI

- **High Risk** - Business Critical test cases.
- Test cases that are **executed repeatedly.**
- Test Cases **that are very tedious or difficult** to perform manually
- Test Cases which are **time consuming**

The following category of test cases are not suitable for automation:

- Test Cases that are **newly designed and not executed manually** at least once
- Test Cases for which the **requirements are changing frequently**
- Test cases which are **executed on ad-hoc basis**.

Automation Process

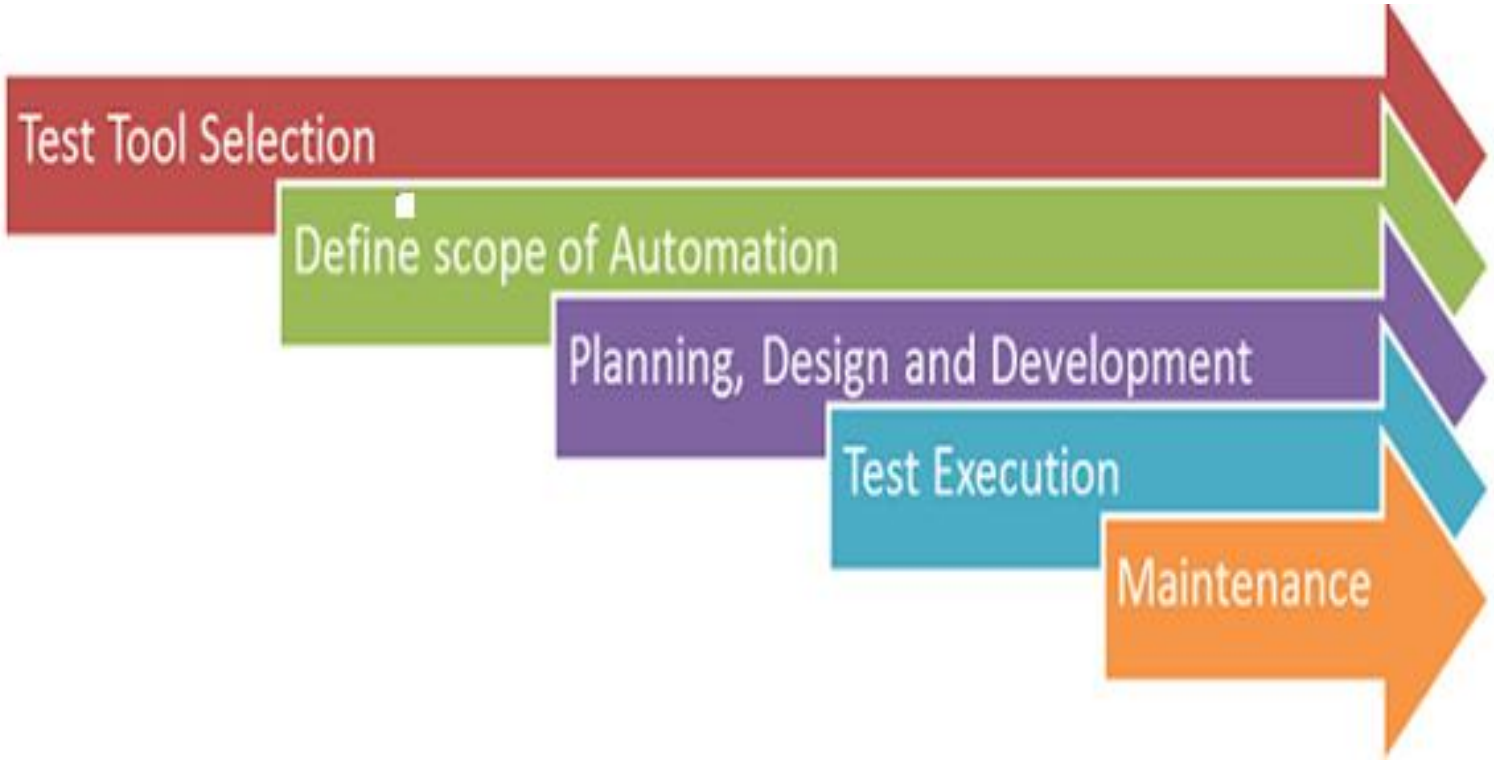


Figure. Automation Processing

1.Test tool selection:

- Largely **depends on** the technology the **Application Under Test** is built on.
- It's a good idea to conduct Proof of Concept of Tool on AUT
- Example:
  - For instance QTP does not support Informatica.
  - So QTP cannot be used for testing Informatica applications

2.Define the scope of Automation:

- Feature that are important for the business
- Scenarios which have large amount of data
- Common functionalities across applications
- Technical feasibility
- Extent to which business components are reused
- Complexity of test cases
- Ability to use the same test cases for cross browser testing

3.Planning, Design and Development

- Automation tools selected
- Framework design and its features
- In-Scope and Out-of-scope items of automation
- Automation test bed preparation
- Schedule and Timeline of scripting and execution
- Deliverables of automation testing

4.Test Execution:

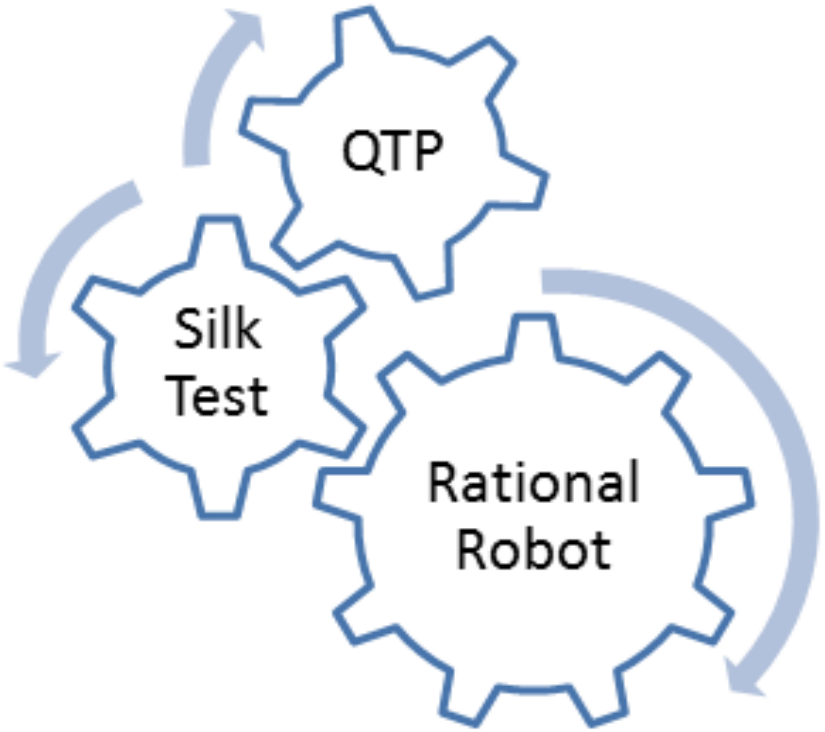
- Scripts are executed during this phase.
- The scripts need input test data before there are set to run.
- Once executed they provide detailed test reports they provide detailed test reports.
- Example: Quality center is the Test Management tool which in turn it will invoke QTP for execution of automation scripts. Scripts can be executed in a single machine or a group of machines.

5.Maintenance:

- As new functionalities are added to the System Under Test with successive cycles, Automation Scripts need to be added, reviewed and maintained for each release cycle.
- Maintenance becomes necessary to improve effectiveness of Automation Scripts.

Automation tools:

- Following are the most popular test tools :



QTP (Quick Test Professional):

- HP's Quick Test Professional (now known as HP Functional Test) is the MARKET leader in Functional Testing Tool.
- Key word driven testing
- Suitable for both client server and web based application
- Better error handling mechanism
- Excellent data driven testing features

Rational Robot:

- Rational Robot is a complete set of components for automating the testing of Microsoft Windows client/server and Internet applications.
- The main component of Robot lets you start recording tests in as few as two mouse clicks.
- After recording, Robot plays back the tests in a fraction of the time it would take to repeat the actions manually.
- Enables defect detection, includes test cases and test management, supports multiple UI technologies

Selenium:

- It is a portable software testing framework for web applications.
- It provides a record/playback tool for authoring tests without learning a test scripting language (**Selenium IDE**).
- It also provides a test domain-specific language to write tests in a number of popular programming languages, including Java, C#, Groovy, Perl, PHP, Python and Ruby.
- The tests can then be run against most modern web browsers.
- Selenium deploys on Windows, Linux, and Macintosh platforms.
- It is open-source software,.

SilkTest:

- **SilkTest** is a tool for automated function and regression testing of enterprise applications.
- It identifies all windows and controls of the application under test as objects and defines all of the properties and attributes of each window. Thus it supports object oriented implementation (Partially).
- It can be run to identify mouse movement along with keystrokes (Useful for custom object). It can use both record and playback or descriptive programming methods to capture the dialogs.
- Extensions supported by SilkTest: .NET, Java (Swing, SWT), DOM, IE, Firefox, SAP Windows GUI.

WinRunner:

- HP **WinRunner** software was an automated functional GUI testing tool that allowed a user to record and play back user interface (UI) interactions as *test scripts*.
- Functionality testing tool
- Supports web technologies such as (VB, VC++, D2K, Java, HTML, Power Builder, Delphe, Cibell (ERP))
- It run on Windows only.
- This tool developed in C on VC++ environment.
- To automate our manual test win runner used TSL (Test Script language like c)

Benefits of Automated Testing

Reliable:
Tests perform precisely the same operations each time they are run, thereby eliminating human error

Repeatable:
You can test how the software reacts under repeated execution of the same operations.

Programmable: You can program sophisticated tests that bring out hidden information from the application.

Comprehensive: You can build a suite of tests that covers every feature in your application.

Reusable:

You can reuse tests on different versions of an application, even if the user interface changes.

Better Quality Software:

Because you can run more tests in less time with fewer resources

Fast:

Automated Tools run tests significantly faster than human users. 70% faster than the manual testing

Cost Reduction:

As the number of resources for regression test are reduced

Automated test tool features:

1. Essential
2. Highly Desirable
3. Nice to Have

1.Essential:

- The ability to divide the script into a small number (3 or 4) of repeatable modules.
- Supports the use of multiple data sheets/tables (up to at least 8).
- The ability to store objects names in the data tables and refer to them in the script.
- The ability to treat the contents of different cells in the data sheets as input data, output data, windows, objects, functions, commands, URL, executable paths, commands etc.
- The ability to access any data sheet from any module.

2.Highly Desirable

- The ability to recover from Severity 1(fatal) errors and still continue to the end.
- The ability for the user to create User defined functions.
- The ability to write directly into the results report.
- The ability to write into the data table (by the script).

3. Nice to Have:

- Direct interface to the test management system (bi-directional).
- The ability to import data sheets “on the fly”.
- The ability to add comments to the data table (rows).
- The ability to restrict output to the results file .

Static Test Tools:

- These are generally used by developers as part of the development and component testing process.
- These tools do not involve actual input and output
- Static analysis tools are an e xtension of compiler technology
- Static analysis tools for code can help the developers to understand the structure of the code, and can also be used to enforce coding standards

Static Test Tools Examples:

- **Flow analyzers:**
  They ensure consistency in data flow from input to output.

- **2) Path tests:**
  They find unused code and code with contradictions.

- **3) Coverage analyzers:**
  It ensures that all logic paths are tested.

- **4) Interface analyzers:**
  It examines the effects of passing variables and data between modules.

Dynamic analysis tools

- **'dynamic'** because they require the code to be in a **running state**.
- They analyze what is happening **'behind the scenes'** that is in the code while the software is running.
- These tools would typically be used by developers in component testing and component integration testing, e.g. when testing middleware, when testing security or when looking for robustness defects.

Features of Dynamic test tools:

- To detect memory leaks.
- To identify pointer arithmetic errors such as null pointers
- To identify time dependencies
- tools test the software system with 'live' data

Dynamic test tools examples:

1) Test driver:

It inputs data into a module-under-test (MUT).

2) Test beds:

It simultaneously displays source code along with the program under execution.

3) Emulators:

The response facilities are used to emulate parts of the system not yet developed.

4) Mutation analyzers:

The errors are deliberately 'fed' into the code in order to test fault tolerance of the system

What is a Testing Framework?

- testing automation framework is an execution environment for automated tests.
- It is the overall system in which the tests will be automated.
- It is defined as the set of assumptions, concepts, and practices that constitute a work platform or support for automated testing.
- It is application independent.
- It is easy to expand, maintain and perpetuate. sting.

Why we need a Testing Framework

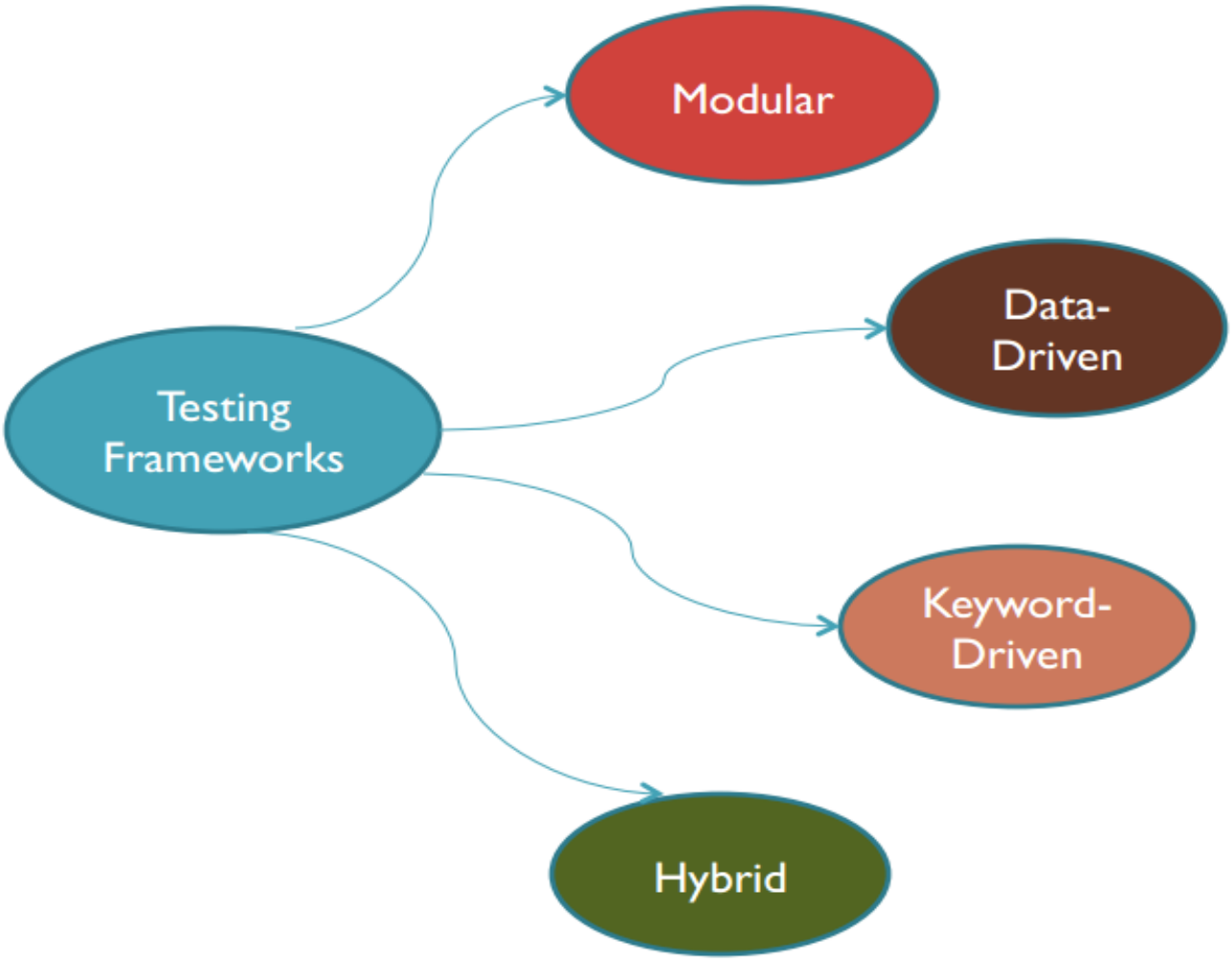
- If we have a group of testers and suppose if each project implements a unique strategy then the time needed for the tester become productive in the new environment will take long.

- To handle this we cannot make changes to the automation environment for each new application that comes along.

- For this purpose we use a testing framework that is application independent and has the capability to expand with the requirements of each application.

- Also an organized test framework helps in avoiding duplication of test cases automated across the application. In short Test frameworks helps teams organize their test suites and in turn help improve the efficiency of testing.

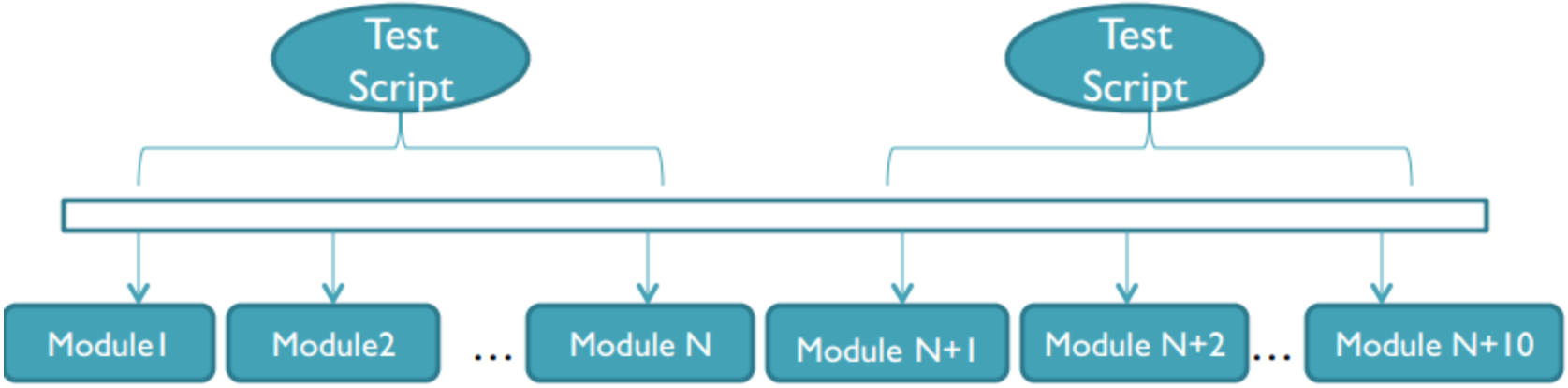
Types Of Testing Frameworks



Modular Testing Framework

- The Modularity testing framework is built on the concept of abstraction.

- This involves the creation of independent scripts that represent the modules of the application under test.



- These modules in turn are used in a hierarchical fashion to build large test cases.

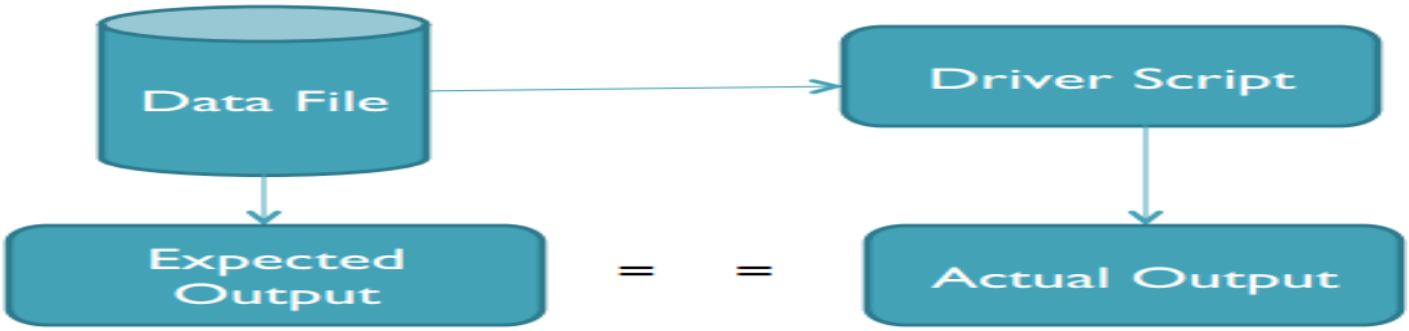
- Thus it builds an abstraction layer for a component to hide that component from the rest of the application.

- Thus the changes made to the other part of the application do not effect that component.

Data-Driven Testing Framework

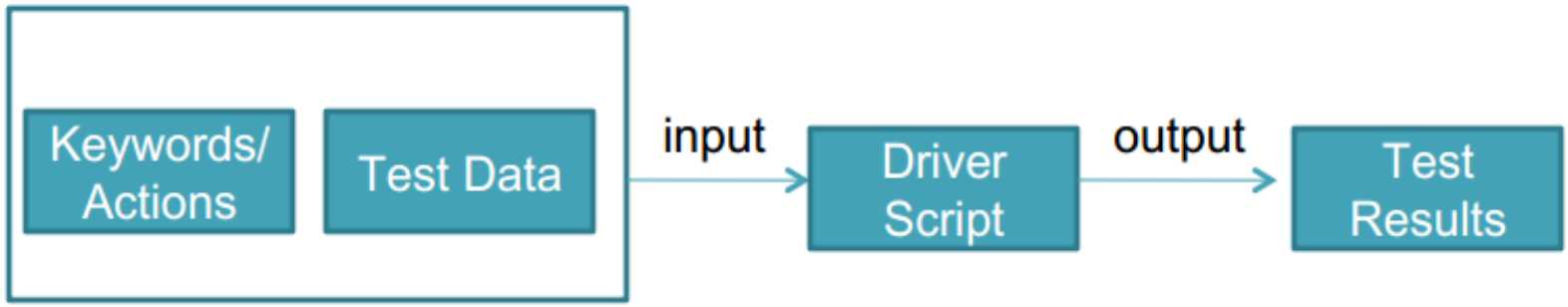
- Data driven testing is where the test input and the expected output results are stored in a separate data file (normally in a tabular format) so that a single driver script can execute all the test cases with multiple sets of data.

- The driver script contains navigation through the program, reading of the data files and logging of the test status information.



Keyword- Driven Testing Framework

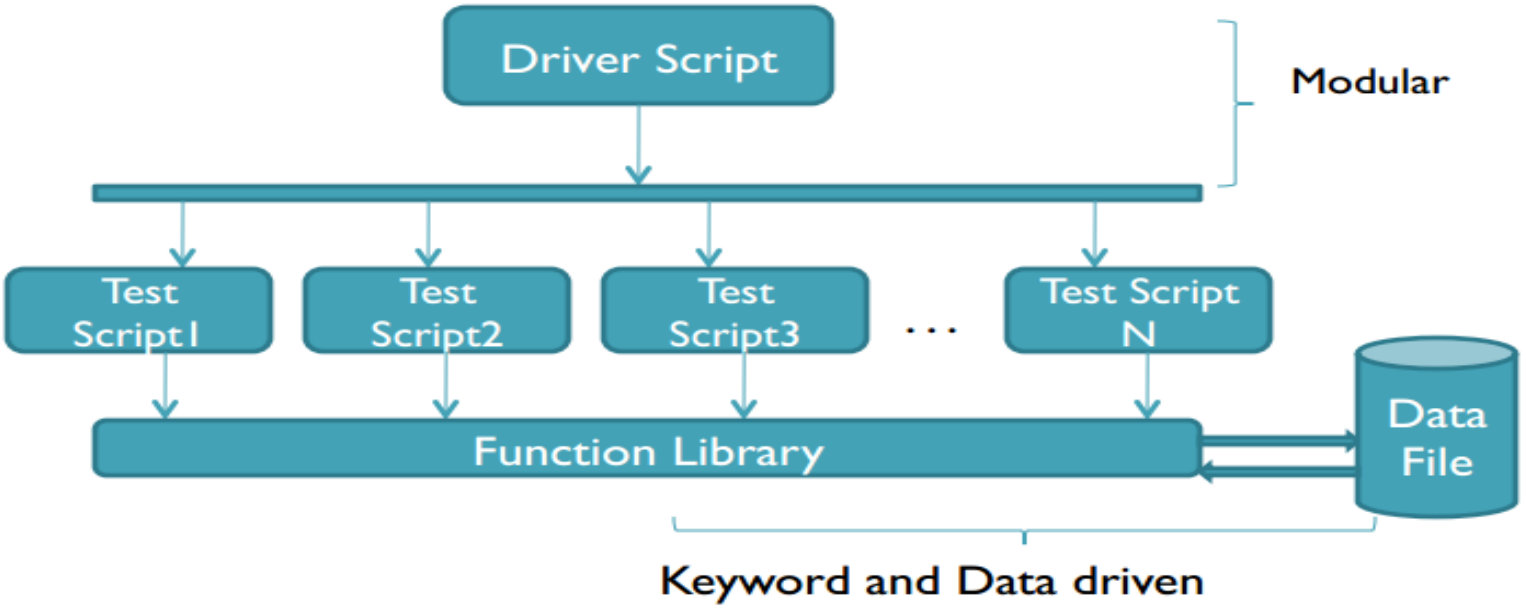
- Keyword driven testing is an application independent framework utilizing data tables and self explanatory keywords to explain the actions to be performed on the application under test.
- Not only is the test data kept in the file but even the directives telling what to do which is in the test scripts is put in external input data file.
- These directives are called keywords.



Hybrid Testing Framework

- Hybrid testing framework is the combination of modular, data-driven and keyword driven testing frameworks.

- This combination of frameworks helps the data driven scripts take advantage of the libraries which usually accompany the keyword driven testing.



Comparison of Frameworks

Approach	Advantages	Disadvantages
Modular testing framework	Modular approach Reusable functions Hierarchical Structure	Test data within the scripts limits reusability, Test script is dependent on software.
Data driven testing framework	Improved Maintainability	Dependency on technical expertise, Test script is dependent on software.
Keyword driven testing framework	Ease of maintenance, Scalability, Less dependency of software.	Dependency on technical expertise, Requires large effort
Hybrid testing framework	Integrates the advantages of all the other frameworks.	Increased Complexity

Software Test Metrics

Metrics can be defined as "STANDARDS OF MEASUREMENT".

- *Metric is a unit used for describing or measuring an attribute.*
- *Test metrics are the means by which the software quality can be measured.*
- *Test provides the visibility into the readiness of the product , and gives clear measurement of the quality and completeness of the product.*

Why we Need Metrics?

- **You cannot improve what you cannot measure.”**
- **“You cannot control what you cannot measure”**

TEST METRICS HELPS IN,

- Take decision for next phase of activities
- Evidence of the claim or prediction
- Understand the type of improvement required
- Take decision on process or technology change

Type of Testing Metrics

1.Base Metrics (Direct Measure)

- Base metrics constitute the raw data gathered by a Test Analyst throughout the testing effort.

- These metrics are used to provide project status reports to the Test Lead and Project Manager; they also feed into the formulas used to derive Calculated Metrics.

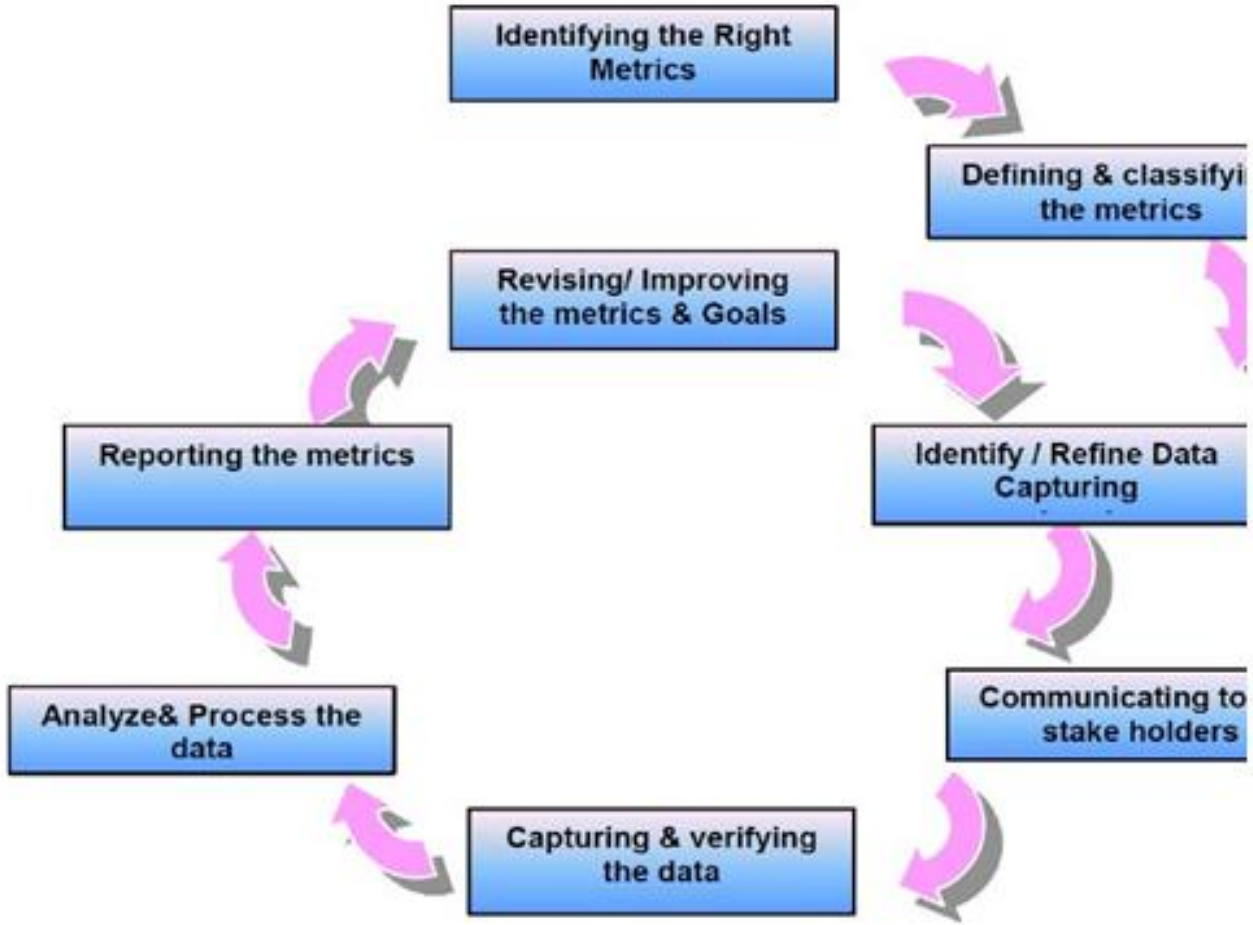
- This data will be tracked throughout the Test Life cycle.

- Ex: no. of of Test Cases, no. of of Test Cases

2.Calculated Metrics (Indirect Measure)

- Calculated Metrics convert the Base Metrics data into more useful information.
- These types of metrics are generally the responsibility of the Test Lead and can be tracked at many different levels (by module, tester, or project).
- Ex: % Complete, % Test Coverage

Metrics life Cycle:



Analysis

- Identification of the Metrics
- Define the identified Metrics.

Communicate

- Explain the need of metric to stakeholder and testing team
- Educate the testing team about the data points need to be captured for processing the metric.

Evaluation

- Capture & verify the data.
- Calculating the metric(s) value using the data captured

Report

- Develop the report with effective conclusion
- Distribute report to the stakeholder and respective representative.
- Take feedback from stakeholder.

Definitions and Formulas for Calculating Metrics:

1) %ge Test cases Executed:

This metric is used to obtain the execution status of the test cases in terms of %ge.

- %ge Test cases Executed = **(**No. of Test cases **executed** / Total no. of Test cases **written**) * 100.

2) %ge Test cases not executed:

This metric is used to obtain the pending execution status of the test cases in terms of %ge.

- %ge Test cases not executed = **(**No. of Test cases **not executed** / Total no. of Test cases **written**) * 100.

3) %ge Test cases Passed:

This metric is used to obtain the Pass %ge of the executed test cases.

- %ge Test cases Passed = **(**No. of Test cases **Passed** / Total no. of Test cases **Executed**) * 100.

#4) %ge Test cases Failed:

This metric is used to obtain the Fail %ge of the executed test cases.

- %ge Test cases Failed = **(**No. of Test cases Failed / Total no. of Test cases Executed) * 100.

- **#5) %ge Test cases Blocked**:

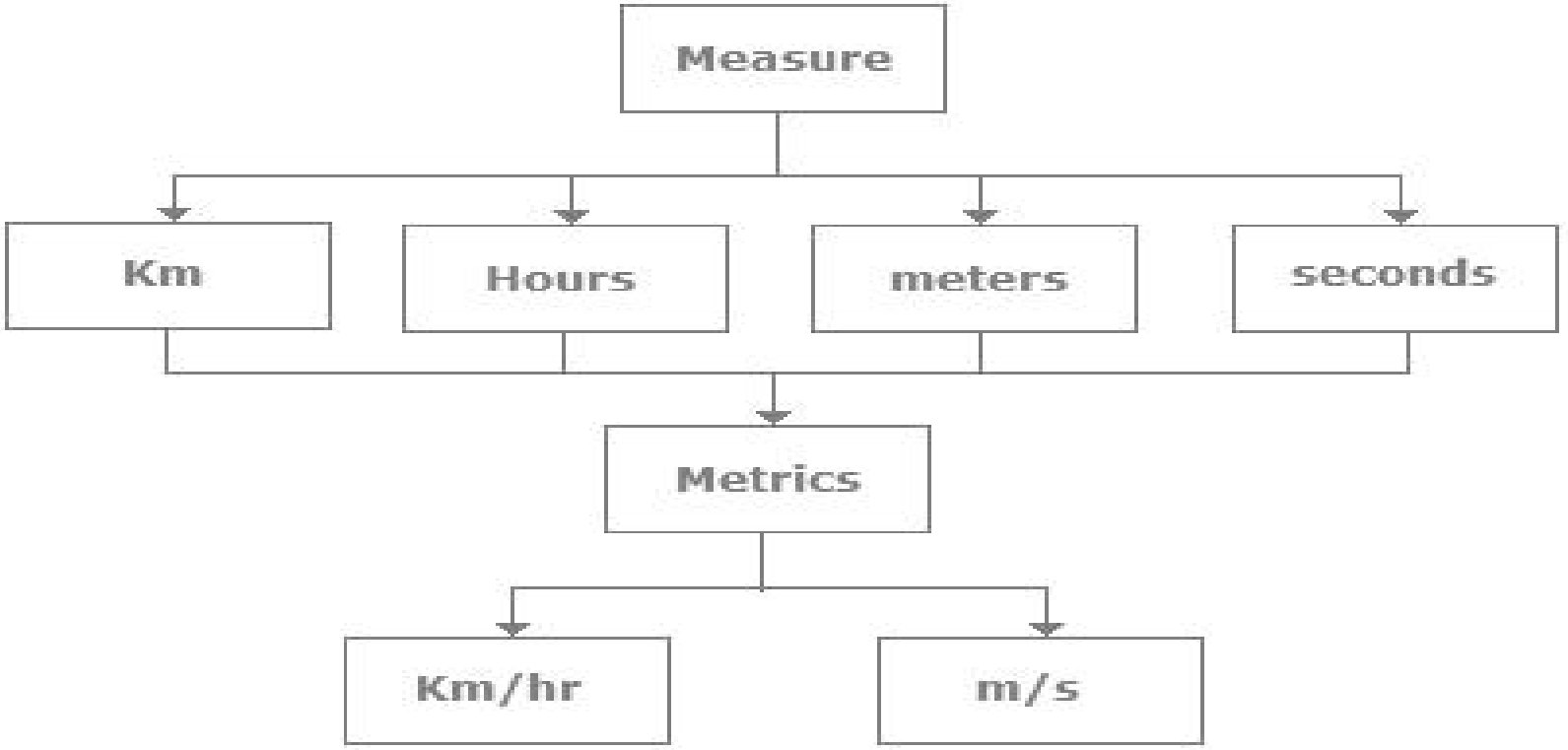
  This metric is used to obtain the blocked %ge of the executed test cases. A detailed report can be submitted by specifying the actual reason of blocking the test cases.

- %ge Test cases Blocked = **(**No. of Test cases Blocked / Total no. of Test cases Executed) * 100.

What is Software Test Measurement?

- *Measurement is the quantitative indication of extent, amount, dimension, capacity, or size of some attribute of a product or process.*
- **Test measurement example:** Total number of defects.

difference between Measurement & Metrics.



Project metrics

- A software team can use software project metrics to adapt project workflow and technical activities.
- Project metrics are used to avoid development schedule delays, to mitigate potential risks, and to assess product quality on an on-going basis.
- Every project should measure its inputs (resources), outputs (deliverables), and results (effectiveness of deliverables).

Project metrics enable project manager to

- Assess status of ongoing project
- Track potential risks
- Uncover problem are before they go critical
- Adjust work flow or tasks
- Evaluate the project team's ability to control quality of software work products

Progress Metrics

- It is the set of metrics to indicate how different activities of the project are processing.
- We used progress metrics to track planned vs. actual over time.
- To provide software quality assurance, we want to track progress of such things as a defects , test cases , man hours.

Productivity metrics

- Productivity is the measure of the efficiency of a process to consume inputs and produce outputs.
- Productivity metrics can be applied to development, enhancement, and support work within IT organizations.
- ***Metric Formula***

$$\text{Development Productivity} = \frac{\text{Development project size (in FP)}}{\text{Development effort (in hours)}}$$

- Development effort is the hours expended doing project work from project inception through to acceptance by the customer or user.

- Development project size is the function point size of the functionality created for the project. Included are conversion and test functions

- Function points are obtained from the organization´s historical project information.