TESTING FRAMEWORKS

Gayatri Ghanakota

OUTLINE

- Introduction to Software Test Automation.
  - What is Test Automation.
  - Where does Test Automation fit in the software life cycle.
  - Why do we need test automation.
- Test Automation using Testing frameworks.
  - What is a testing framework.
  - Why do we need a testing framework.
  - Types of testing frameworks.
  - Comparison of different frameworks.
- Shift from Waterfall to Agile.
- Test Driven Development and Behavior Driven Development.
- Summary

Introduction To Software Test Automation

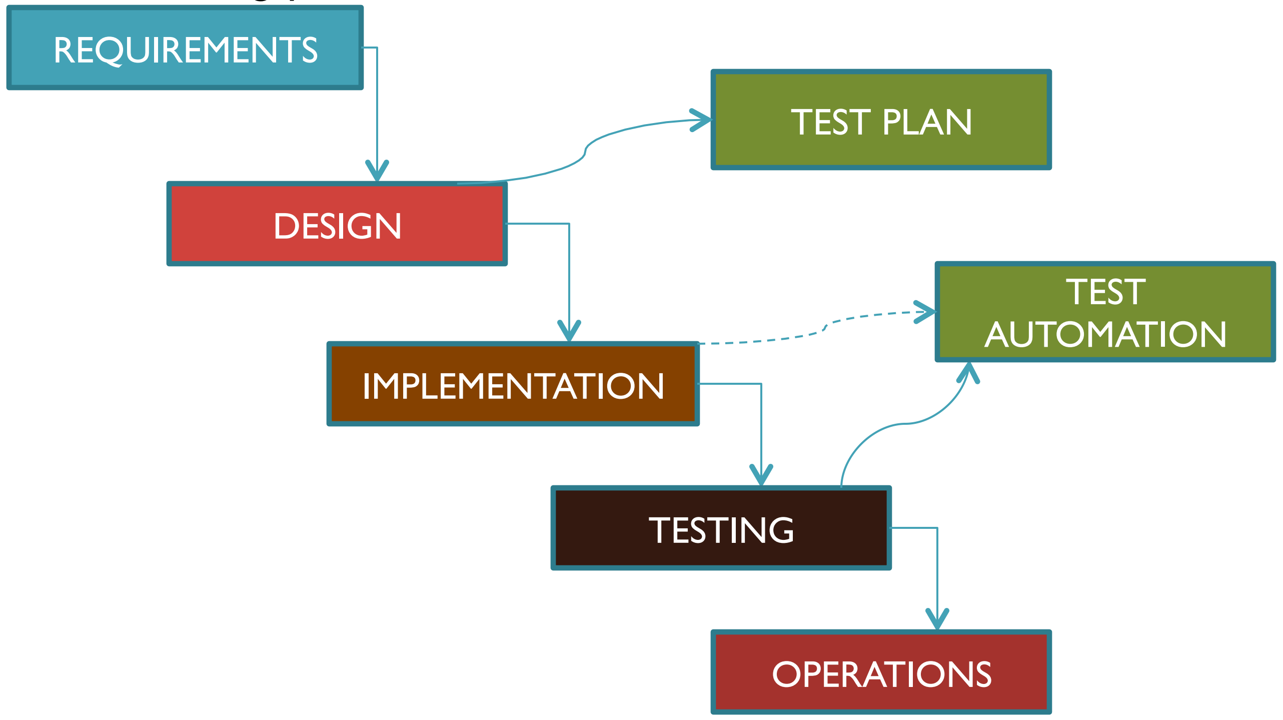
What is Test Automation

- Test automation is the use of software (under a setting of test preconditions) to execute tests and then determine whether the actual outcomes and the predicted outcomes are the same.

- For example, Windows Vista offers per-application volume. It is possible to turn down the volume on a game while leaving Windows Media Player playing loud. To do this, right-click on the speaker icon in the lower-right-hand corner of your screen and select "Open Volume Mixer." Moving the slider for an application down should cause its volume to decrease. Testing this manually is easy. Just play a sound, lower the volume, and listen. Now automating this rather than doing it manually is the process of test automation.

Where does Test Automation fit in the Software Life Cycle

- Considering the earlier software life cycles such as the waterfall model the test automation appears in this life cycle during the implementation and testing phase.



Why do we need Test Automation

- Companies not only want to test software adequately, but also as quickly and thoroughly as possible. To accomplish this goal, organizations are turning to automated testing.
- To increase the test coverage
- Reduces the need for manual testing and discovers defects manual testing cannot expose and also manual testing is error prone and a time consuming process.
- Running the tests again and again gives us the confidence that the new work we added to the system did not break the code that used to work and also to make sure that the changes we introduced are working.
- Executing the tests (particularly acceptance tests) can also help us understand what portion of the desired functionality has been implemented.

- The set of the automated test suite can form a regression test suite. The purpose of the regression suite is to make sure that the software behavior is unchanged unless due to data change or latest software.
- Automating also reduces the time taken for regression testing.
- Automated unit test suite helps find the problems at an earlier stage and solve them.

Test Automation Using Testing Frameworks

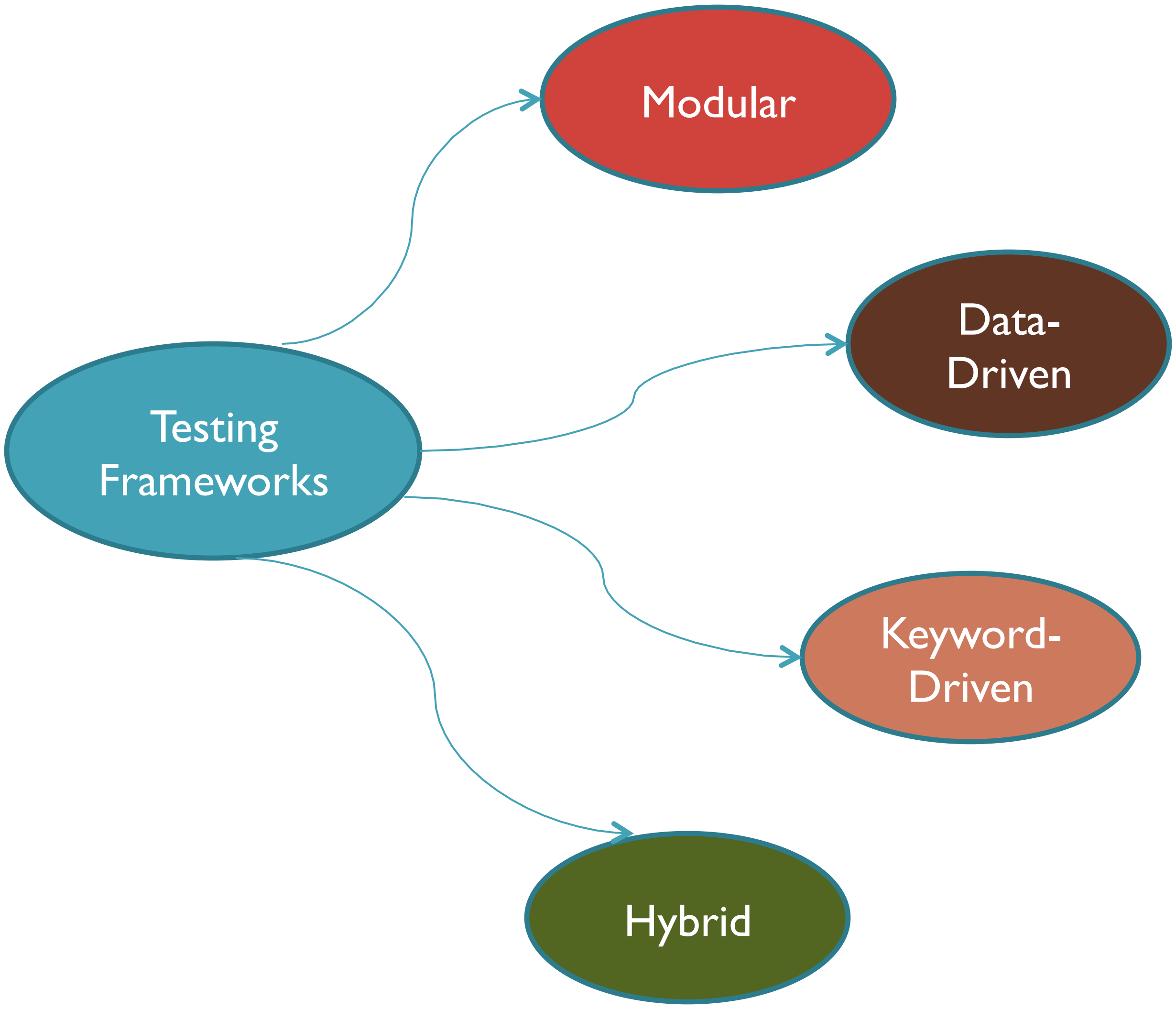
What is a Testing Framework

- A testing framework or more specifically a 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.
- The Testing framework is responsible for:
    - Defining the format in which to express expectations.
    - Creating a mechanism to hook into or drive the application under test
    - Executing the tests
    - Reporting results
- Properties of a testing framework:
    - It is application independent.
    - It is easy to expand, maintain and perpetuate.

Why we need a Testing Framework

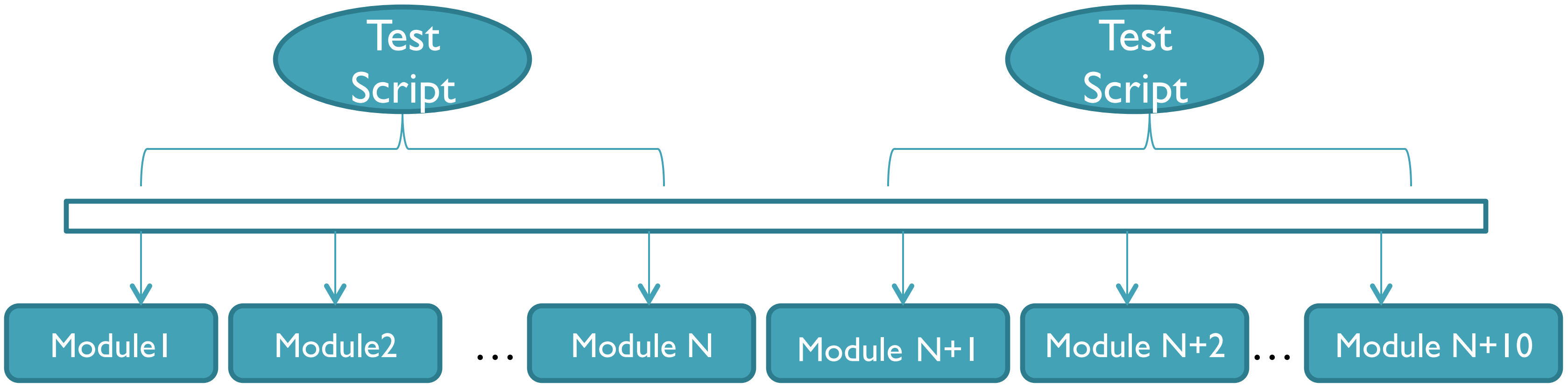
- If we have a group of testers and suppose if each project implements a unique strategy then the time needed for the tester become productive in the new environment will take long.
- To handle this we cannot make changes to the automation environment for each new application that comes along. For this 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.



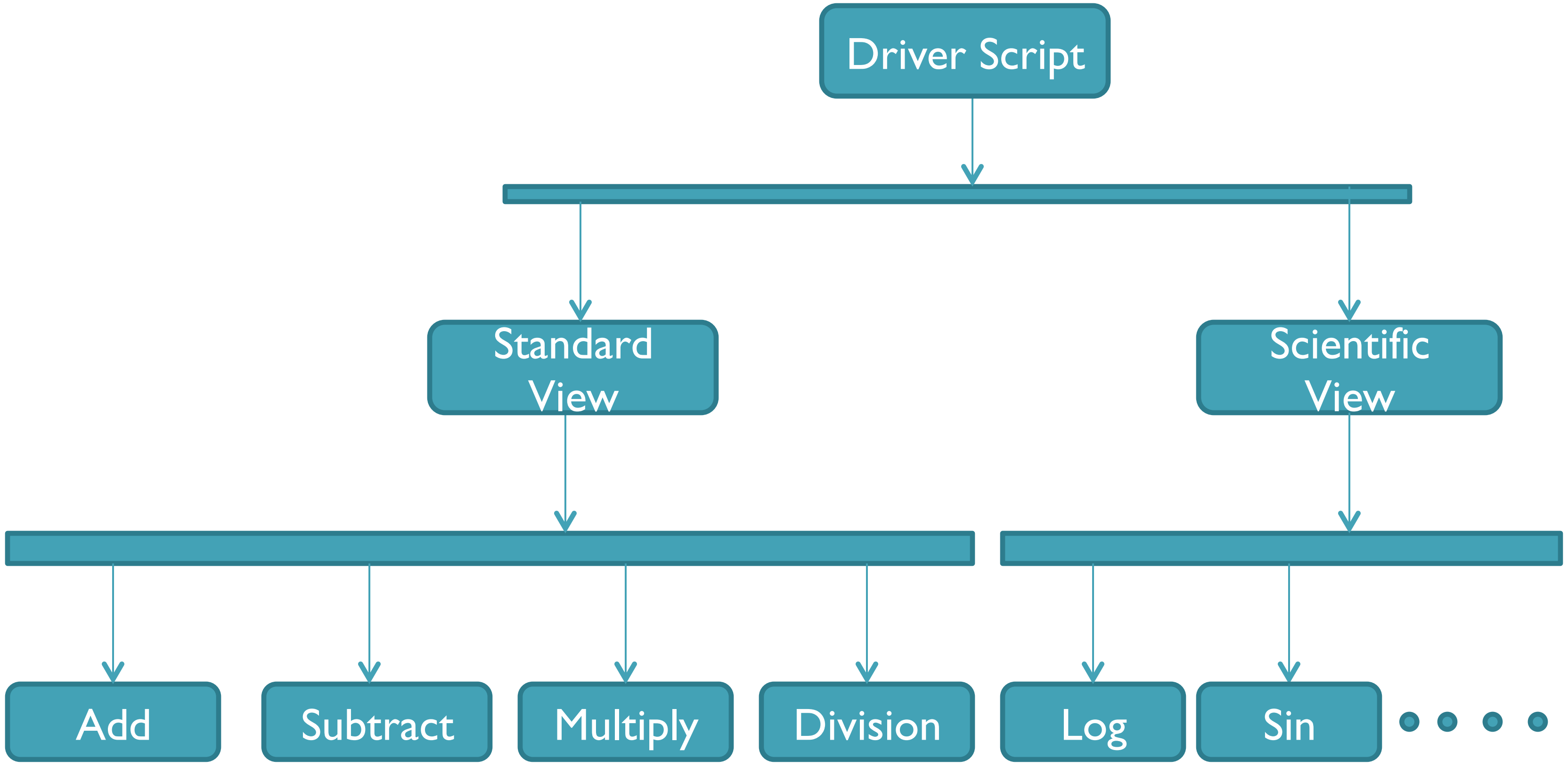
Example of Modular Testing Framework

- To demonstrate the modular framework we use the calculator program.
- Consider the basic functions of the calculator such as addition, subtraction, multiplication, division which are part of the Standard view.
- We create scripts for these functions as follows:

Add:

```
Sub Main
    Window Set Context, "Caption=Calculator", "“
    PushButton Click, "ObjectIndex=10“                          ‘Press 5
    PushButton Click, "ObjectIndex=20“                          ‘Press  +
    PushButton Click, "ObjectIndex=14“                          ‘Press  6
    PushButton Click, "ObjectIndex=21“                          ‘Press =
    Result = LabelUP (CompareProperties, "Text=11.", "UP=Object Properties")
                                                                ‘Compare Expected to Actual Results
End Sub
```

- In a similar way we create scripts for subtraction, multiplication and division.
- At the next level of hierarchy, we create two scripts for standard view and scientific view of which the standard view contains calls to the scripts we created as before.



The Driver script is the top most level of hierarchy which contains the scripts of standard and scientific view.

Driver Script:

```
Sub Main

    'Test the Standard View
    CallScript "Test Script Mod Framework - Standard"

    'Test the Scientific View
    CallScript "Test Script Mod Framework - Scientific“

End Sub
```

Thus this framework introduces a high level of modularization. So when there is a change in the functionality we can change the bottom level script without effecting all the other test cases that test that control.

Modular Testing Framework - Contd

Advantages:

- Modular division of scripts leads to easier maintenance and also the scalability of the automated test suites.
- The functionality is available in easy to use test libraries so creating new driver scripts for different tests is easy and fast.

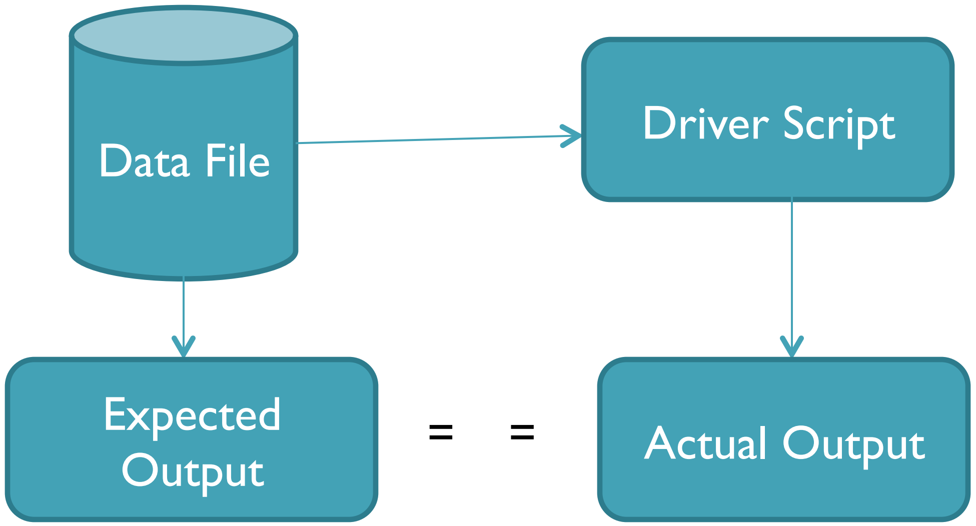
Disadvantages:

- The main problem with modular frameworks is that the test script have test data embedded in them. So when the test data needs to be updated we need to change the code of the script. This becomes a big problem when the test script is large.

  For this purpose, data- driven testing frameworks have been introduced.

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.



Example of Data Driven Testing Framework

- To demonstrate the data driven testing framework we use the login page of the flight reservation
- The first step involves creating the test data file. (testdata.csv)

Test Case	Number1	Operator	Number2	Expected Result
Add	2	+	3	5
Subtract	3	-	2	1
Multiply	2	*	3	6
Divide	2	/	-2	-1

- This data file contains the different types of input data which will be given to the driver script.

- In the next step we create a driver script and make references to the test data file.

```
data = open ( ’ testdata.csv’ ) . read ( )

l i n e s = data . s p l i t l i n e s ( )  #excluding the header row

for line in lines:

        Read Number1

        Read Number2

        Read Operator

        Calculate the result using the Operator on

        Number 1 and Number2

        Compare the result to the expected result
```

- This driver script reads the data from the data file computes the value and compares it with the expected result from the data file.

Data-Driven Testing Framework - Contd

Advantages:

- This framework reduces the number of overall test scripts needed to implement all the test cases.
- Less amount of code is required to generate all the test cases.
- Offers greater flexibility when it comes to maintenance and fixing of bugs.
- The test data can be created before test implementation is ready or even before the system to be tested is ready.

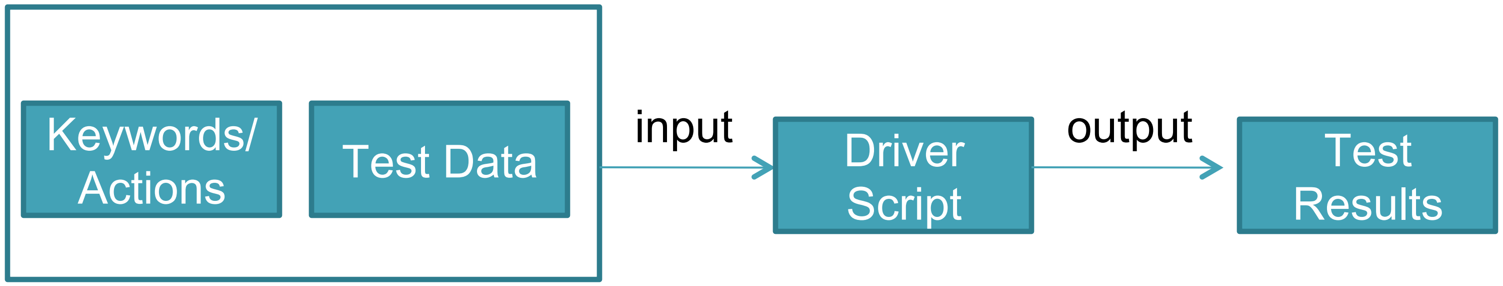
Disadvantages:

- The test cases created are similar and creating new kind of tests requires creating new driver scripts that understand different data. Thus the test data and driver scripts are strongly related that changing either requires changing the other.

For this purpose keyword driven testing frameworks have been introduced.

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. The keyword based testing is an extension to the data driven testing.



Example of Keyword Driven Testing Framework

- To demonstrate the keyword driven testing we take the actions performed by the mouse when making calculations.
- We create a table that maps the actions performed with the mouse on the window of the calculator application. In this table,
    - The windows column represents the application for which we are performing the mouse action.
    - The control column represents the control that we are clicking with the mouse.
    - The action column represents the action performed by the mouse.
    - The argument column contains the specific control value.

Window	Control	Action	Arguments
Calculator	Menu		View, Standard
Calculator	Pushbutton	Click	2
Calculator	Pushbutton	Click	+
Calculator	Pushbutton	Click	3
Calculator	Pushbutton	Click	=
Calculator		Verify Result	5
Calculator		Clear	
Calculator	Pushbutton	Click	5
Calculator	Pushbutton	Click	-
Calculator	Pushbutton	Click	3
Calculator	Pushbutton	Click	=
Calculator		Verify Result	2

- After creating the table, we create a set of scripts for reading in the table, executing each step based on the keyword contained in the action field and logs any relevant information.
- The below pseudo code represents this test of scripts.

```
Main Script / Program
          Connect to data tables.
          Read in row and parse out values.
          Pass values to appropriate functions.
          Close connection to data tables.
Return
Menu Module Set focus to window.
          Select the menu pad option.
Return.
Pushbutton Module Set focus to window.
          Push the button based on argument.
Return.
Verify Result Module Set focus to window.
          Get contents from label.
 Return
Compare contents with argument value.
          Log results.
Return.
```

Keyword- Driven Testing Framework

Advantages:

- It has all the advantages that data driven testing has.
- Automation expertise is not required to maintain or create a new set of test cases.
- Keywords are reused across multiple test cases.

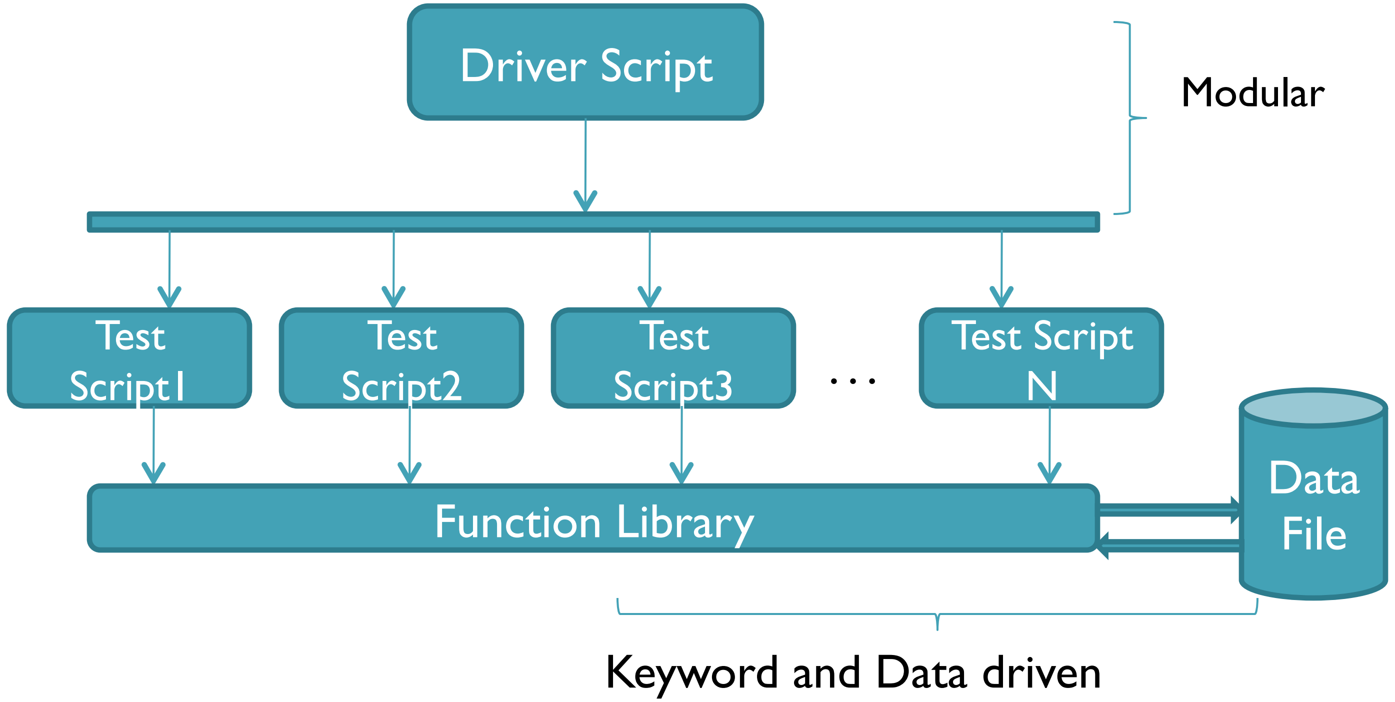
Disadvantages:

- The main problem is that this requires a more complicated framework than the data driven framework.
- With the keyword driven approach the test cases get longer and complex and this is due to the greater flexibility that this approach offers.

So in order to combine the strengths of all the frameworks and mitigate their weaknesses we use the hybrid testing framework.

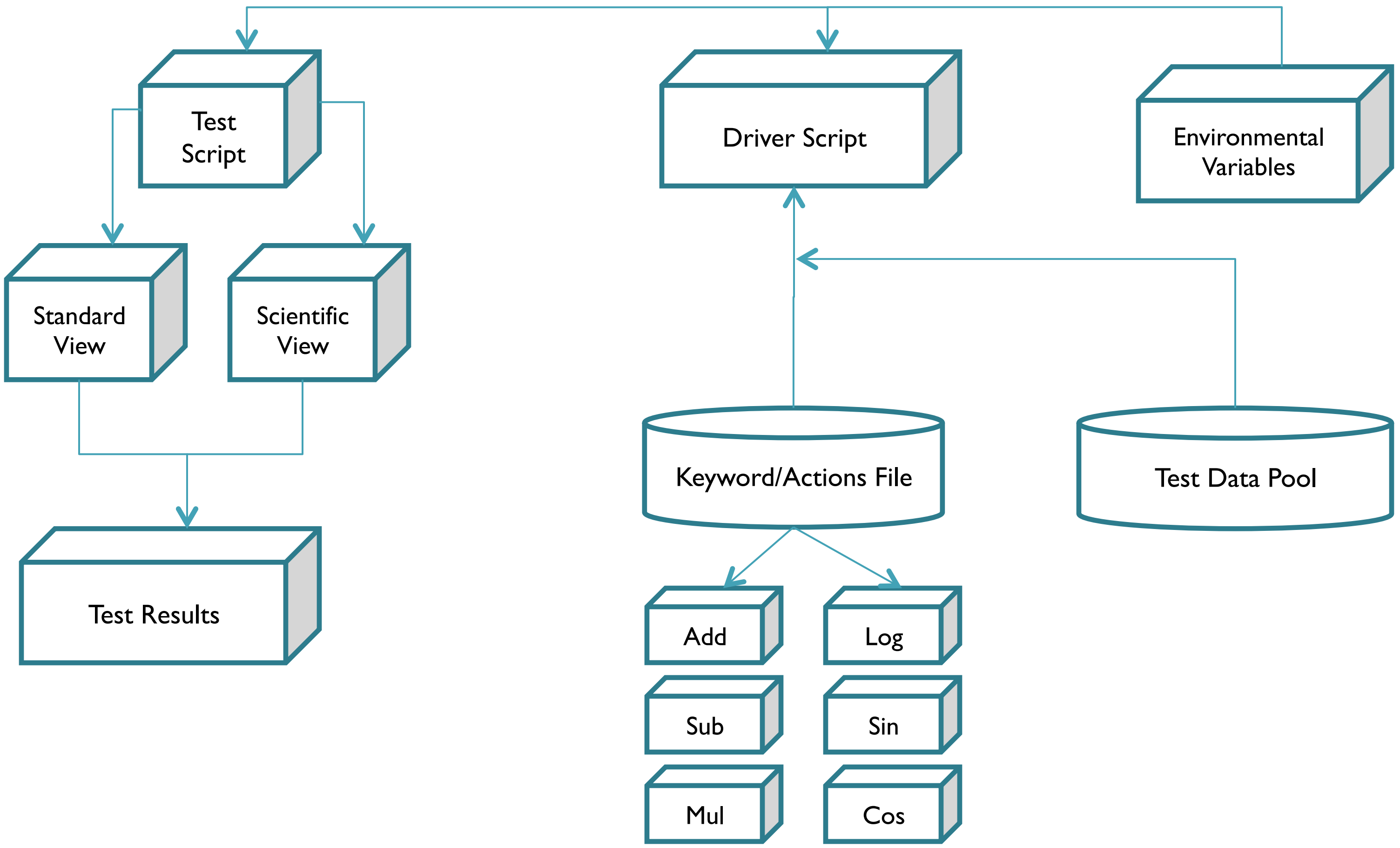
Hybrid Testing Framework

- 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.



Example of Hybrid Testing Framework

The hybrid framework for the calculator can be shown as follows:



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

Shift from Waterfall to Agile

- We do not see a working version of the software until late in the waterfall life cycle. Problems may be more costly to fix in this phase than they would have been earlier in the life cycle.
- Using test automation in the water fall model with feedback does not have many advantages as only regression testing is covered in the test automation and every time regression testing of the previous version has to be executed.
- Hence it is required to start test automation early in the software development life cycle.
- Test automation with agile methodologies has advantages compared with the traditional life cycles as testing is done throughout the life cycle process.
- In the agile life cycle the test automation execution starts early in the software life cycle.

Testing Frameworks in the Agile

- Agile life cycles are characterized by short and rapid test cycles and frequent change requests. Thus test automation plays a crucial role in software testing.
- Any type of testing framework can be implemented in the agile environment but with the short iterations and rapidly changing requirements it becomes difficult to maintain the test automation suite.
- In the agile environments, testing plays a crucial role through the different phases of iterations. It involves continuous integration, unit testing (which is usually done using test driven development) and constant regression testing which is difficult to accomplish using testing frameworks.
- Also, achieving maximum code and functionality coverage using testing frameworks is difficult.

Hence testing frameworks are not a good fit for the agile environment.

Test Driven Development
&
Behavior Driven Development

- Test driven development is a technique of using automated unit tests to drive the design of software and force decoupling of dependencies.

  With traditional testing a successful test finds one or more defects. But using TDD we have a clear measure of success when the test no longer fails. Thus TDD increases our confidence that the system meets the requirements and that the system is working properly when compared to the confidence that traditional testing provides.

Why TDD?

- To avoid wasting time on debugging.
- To improve the quality of code.
- To increase confidence.

- Behavior driven development is an extension to the test driven development in that it focuses on the behavior of the system rather than the implementation aspect of the system. Thus it gives a clear understanding of what the system should do to both the developer as well as the customer making the testing process even more efficient.

Why BDD?

- Promotes Outside-In Development.
- BDD = TDD + automated acceptance testing.

Summary

- The test framework should be application-independent.
- The test framework must be easy to expand, maintain, and perpetuate.
- Data driven testing is the quickest and easiest to implement if we have a technical expertise.
- Keyword driven testing is the hardest and most time consuming but once implemented it is the easiest to maintain.
- Hybrid testing combines the advantages of all the other frameworks but requires technical expertise and is useful for long term projects.
- With agile methodologies, test driven development and behavior driven development are more useful as they ensure testing of the application to the fullest.

References

- http://safsdev.sourceforge.net/Default.htm
- http://www.ibm.com/developerworks/rational/library/591.html
- http://msdn.microsoft.com/en-us/library/ff649520.aspx#mtf_ch02_softwaredevelopment
- http://eliga.fi/Thesis-Pekka-Laukkanen.pdf
- http://www.agiledata.org/
- http://www.rimtengg.com/iscet/proceedings/pdfs/se/77.pdf