

Steps to Develop Your Own Framework

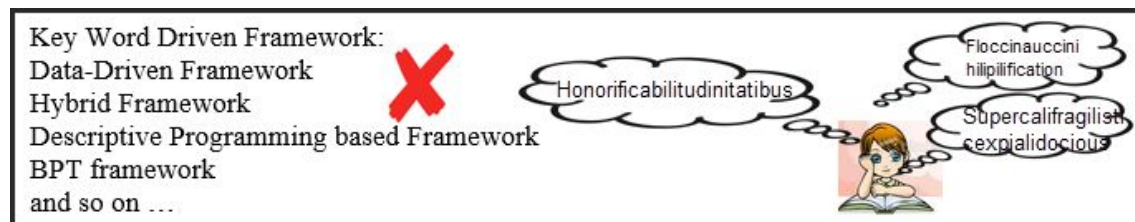


“You Can Call Us Autobots”

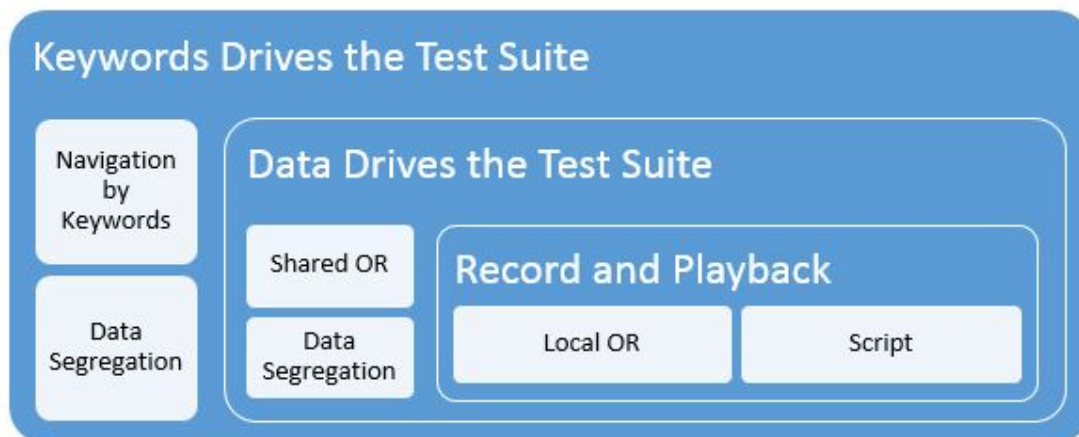
Every task bound to have some kind of process though process can be ad-hoc or very sophisticated to execute same procedure. For example, suppose we want to create script then the following process may follow 1. Analyse application 2. Record 3. Modify 4.Exception handling 5. Archive important data 6. Execute 7. Share results to all stakeholders 8. Modify script for any change 9. Maintain data

These processes need to repeat by every team member. Team member's discretion may make the work product variable across the team. Mature process limit this variable. Suppose if process says to declare all the variable at beginning of the script then any undeclared variable will not able to introduce any bug.

Framework, in broader sense, conceptualize the way of doing thing, organization of components, standardization of communication for inter/intra components and facilitate a complete end-to-end engineering. There are many jargon associated with frameworks word in UFT world. But framework can be seen as the organization of components to accomplish workflow. So, basically it is set of guideline to suggest the way to do things. You may encounter the various framework name during your learning. The common known frameworks in UFT fraternity are



And can be translate in the following concept.



But you should not look the framework with these jargon. You should think in the way of test-architect. The framework should develop based on your testing need. Framework should able to achieve the goal and objective of good framework. Following are the common goal for good framework:

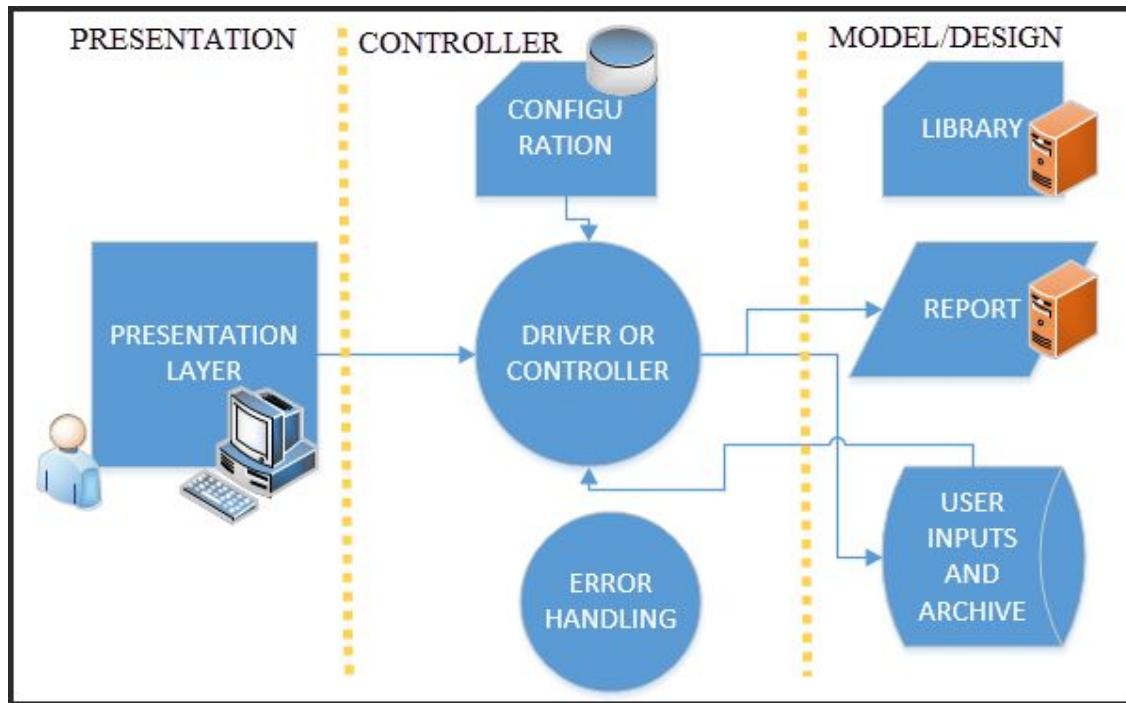
	Characters	Detail	Example
1	Maintainable	Change in application, data or system settings should not force the major rework in test automation.	Sign-In button changes to login button. Change requires in all 100 scripts

2	Less code to write	Avoid unnecessary code by introducing best practice of coding standards, naming convention, loops etc. Use best algorithm and modular code.	Loops, Unnecessary excel connection
3	Consistent	Consistent standard should follow by everyone. Walkthrough of sample code should reveal the big picture	Naming convention, Global File names, Global Input and output data, Global exception handling
4	Reliable & Robust	The automation suite should not fail for small errors. It should be robust to accommodate changes	Array bounds, File formats, Database drivers and so on.
5	Scalable	Adding the component or increasing the coverage of test should be smooth	Adding payment facility (e.g.. debit card) should be accommodate in framework
6	Generic solutions	Framework should comprise generic functions for the task required across the test automation test suite.	ADODB connections, COM api reusable code, FSO, dictionary, recovery and Global functions
7	Platform Independent	Framework should not stuck to one machine or one particular version of system. It should work till the limitation of tool	Framework implement on Window XP and Vista but unable to run on Window7 onwards
8	Cost Efficient	Implementation and execution cost should not be exuberant	Return on Investment analysis should be given its due preference.
9	Secure	Framework should not reveal any data, application bug areas	Test artefacts like files, scripts, reports and component should be version control with appropriate permissions.
10	Performance Efficient	Framework should not deter by the errors and execute with in expected duration	Object memory management, Recover management, running mode.
11	3rd Party tool Integration	Framework should able to accommodate external tools and plug-ins.	Quality Center, Jira and other tools should able to club with test suite.

We wish to design the framework where users can do their part in the process of creating an automation suite. For example business analyst or manual testers can create test cases, administrator can control the infrastructure assets(servers, databases), test engineers can work on the maintenance and updating of test framework and test architect can work on framework improvement, feature enhancement and scale on other platforms and applications.

So, we can view our activity in following manners. Presentation layer gives opportunity to create test case, modify test definition, choose script in test suite, location and other user customize options. This layer can be built in any technology. The objective of this layer to understand BA or testers.

Controller layer is internal layer that influence by the configuration and run the test suite. Design/Model layer is the repository layer. This layer is contains reusable code, data, historical information, reusable repositories. These all layers can be develop simultaneously and can have a span of multiple machines, data sources and environments.



Let's explore our framework development journey to achieve these goal.

UFT records all the steps in script with hard-coded data and creates a local repository. This can be considered as the basic framework. Wow! So, effectively we haven't put any brain inside the test automation and become the framework architect. But it is highly unsatisfied accomplishments. But I hope you understand by now that Framework is "kind-of agreement" of doing things. They need not to be overwhelmed by techno-glossy-array. Record and play back is good when we need to learn about application, test application few times only or fill the form data iteratively.

Now let's analyse what is the problem with record and playback. Let's analyse the recorded script:

```
SystemUtil.Run "iexplore.exe"
Browser("Browser").Navigate "http://quicktestpro.co.uk/"

Browser("Browser").Page("Quick Test Pro").WebEdit("UserID").Set "Mahavir"
Browser("Browser").Page("Quick Test Pro").WebEdit("UserID").Set "Meena"
Browser("Browser").Page("Quick Test Pro").WebButton("SignIn").Click

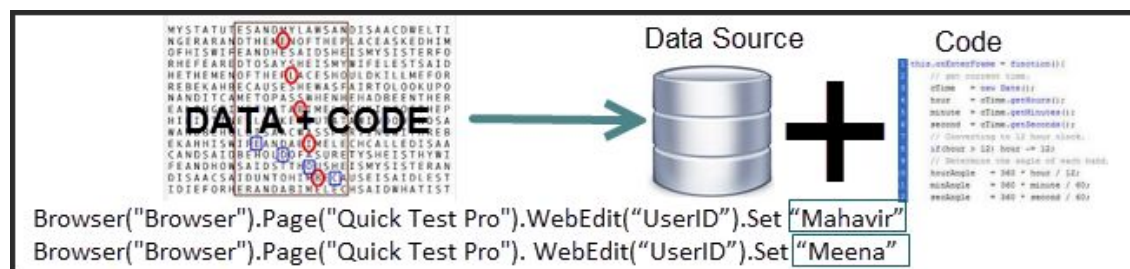
Browser("Browser").Page("Quick Test Pro").Link("Home").Click
Browser("Browser").Page("Quick Test Pro").WebButton("LogOut").Click
Browser("Browser").Page("Quick Test Pro").Sync
Browser("Browser").CloseAllTabs
```

	Test STEPS	Possible Problems (You can think many more)
1	SystemUtil.Run "iexplore.exe"	Other browser session already open Browser crashes during opening IE Running test on other machine where browser is not present
2	Browser("Browser").Page("Quick Test Pro").WebEdit("UserID").Set "Mahavir" Browser("Browser").Page("Quick Test Pro"). WebEdit("UserID").Set "Meena" Browser("Browser").Page("Quick Test Pro").WebButton("SignIn").Click	User id is hard coded and it can run for one user only Password is visible to everyone Sing In happens nearly in every script – Repeated code If any object changes, all script performing login fails.
3	Browser("Browser").Navigate "http://quicktestpro.co.uk/"	Hardcoded URL. Same test cannot run on .com, .net or other site
4	Browser("Browser").Page("Quick Test Pro").Link("Home").Click	If "Home" Object change this script will fail. We can fix it. But what if this step used in 100 script. Is that really we born for? To open and modify every script?
5	Browser("Browser").Page("Quick Test Pro").WebButton("LogOut").Click	If logout not happen properly, session remains open. It will create issue during next login.
5	Browser("Browser").Page("Quick Test Pro").Sync	This step need to append in each test - Unnecessary code
6	Browser("Browser").CloseAllTabs	This step need to append in each test - Unnecessary code


The gist of above problem finding exercise is that we are unable to control the test flow, data, errors and lines of code. To rectify these issues we should implement strategy to weed out these design flaws from our framework architecture.

STEP1: Let's make our script independent of data – Maintainability

If you noticed, script contains the data associated with script. Our goal is to make it independent of data



This data source can be anything. It should correlate with your testing needs e.g. website URL/env, CONST is most appropriate while global data across all the test environment variable proves crucial.



Data Source	JumpStart
Variable	Dim username
Constant	CONST username
Datatable	Datatable("userName","Global")
Excel	CreateObject(Excel.Application) CreateObject("ADODB.connection")
CSV	CreateObject("Scripting.FileSystemObject")
TextFiles	CreateObject("Scripting.FileSystemObject")
Database	CreateObject("ADODB.connection")
Dictionary	CreateObject("Scripting.Dictionary")
Action Parameter	Parameter("userName")
Test Parametr	TestArgs("userName")
Environment Variable	Environment.value("userName")
XML Files	XMLUtil.CreateXML()
Random Number	randomnumber(1,10) Rnd with Randomize
3rd Party Tool Data	CreateObject("TDApiOle80.TDConnection")
User Data	InputBox("Please provide the user Name")

So, now may you feel that framework is not define set of entity but it is our thinking to solve the design problem? The datasource list can go much long if you think beyond the UFT world for example you can pipe the stream data, business intelligence data, Screen readers, Data generated with 3rd party components and so on.

STEP2: Modularize the code – Reusability

Functions, shared OR and Actions provide the way to modularize the code. Functions/Subprocedure can be used multiple time and can be associate during run time or design time in UFT. Now in our example we have login code this code can be bind in a function

Function Login (userName, userPassword)

Browser("Browser").Page("Quick Test Pro").WebEdit("UserID").Set userName

```
Browser("Browser").Page("Quick Test Pro"). WebEdit("UserID").Set userPassword
Browser("Browser").Page("Quick Test Pro").WebButton("SignIn").Click
```

End Function

In same way we can develop the reusable object repositories and actions.

Shared OR	Functions	Data
Scripts-1, Script-2... Script-n		

STEP3: Test Suite – Organization

In previous section we found that test suite was not there. We need to select each script independently and run it exclusively. We should provide a way to organize our test scripts in test suite where user can select and run the specific slot of scripts (Script1 and Script3).

Shared OR			Test Automation Suite		
Functions	Data		Script1	Run	Functions, OR,Data
Scripts-1, Script-2... Script-n			Script2	NoRun	
			Script3	Run	

Now the question is how we can organize the front end where we can select the script. There are various ways that can provide the user interface to select the test suite e.g. Webportal, window applications, batch file, UFT batch runner, .csv files and excel application.

Web Application
PHP,ASP,JAVA...

	A	B	C	D
1	TestCase	IsIncluded	Path	Parameters
2	TC_01	Run		
3	TC_02	NoRun		
4	TC_03	Run		

Form1

Type Here

TC_01 ☒ Window Applications
C#, VB...

TC_02 ☒

TestCaseIs,IncludedPath,Parameters
TC_01,Run
TC_02,NoRun
TC_03,Run

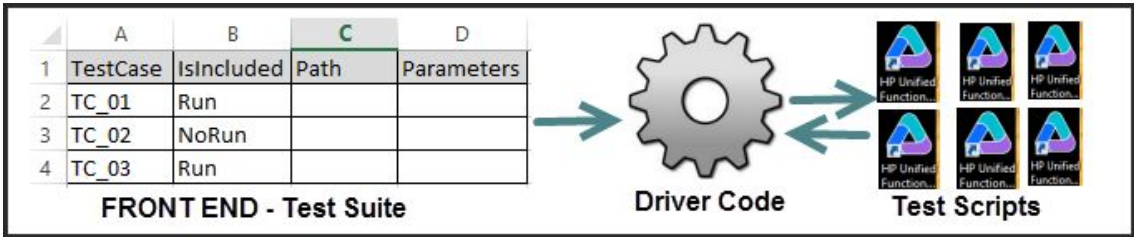
Comma Separated .csv

BAT file to run batch

```
Microsoft Windows [Version 6.2.9200.21713]
(c) 2012 Microsoft Corporation.
C:\Users\Krish>TestSuite.bat
```

These are just example and you can think many more ways to interact with test scripts. The only important thing is that you should be able to communicate with front end and scripts. So you may need

driver for that. Driver is a program to make connection with front end and speak with business flow (can loosely understand as scripts but there can other including descriptive programs, reusable function and so on).



STEP4: Pull out Navigation from Script – Flexibility and Scalability

As of now we have witnessed the design of test suite and reusable components like OR and library. But our script still comprise the flow in terms of steps. We can take out these steps and build our own flow. Flow can have various granularity ranging from object level to function level.

Navigational segregation from Test scripts	
Function Level Granularity	Object Level Granularity
Login	WebEdit:=userid ; WebEdit:=password;WebButton:=signin Operation "Click"
Search	WebEdit:=Search <Search data from Source> Operation "Set"
Buy	WebSelect:=Buy <Pay data from Source> Operation "Click"
Logout	WebLink:=Logout Operation "Click"

We can device the function levels with help of class or simple UFT functions as well. ClassName.Function name where class can be logical division or page based division.

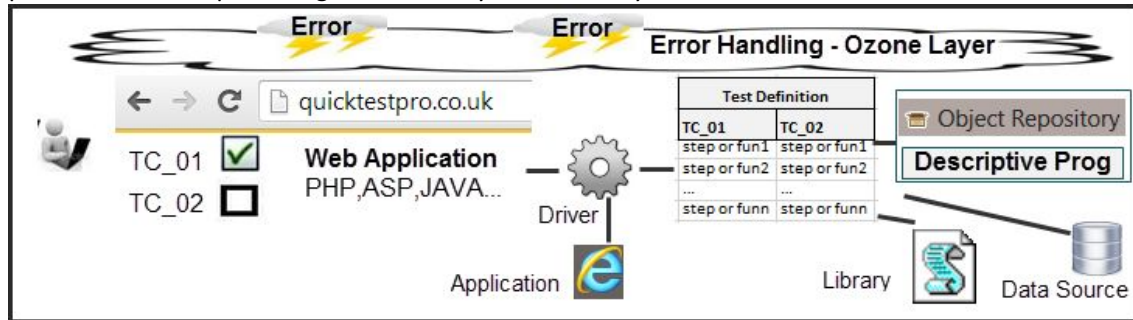
So, in this way user can build their own test cases without going in to the test script. The only need is to manage functional library or object description.

STEP5: Error handling Umbrella on the overall framework- Reliable & Robust

UFT supports VBS and recovery scenario. We should mix both of the strategy to manage the exception at global and script level. It is one of the most important things to incorporate in the framework. Recovery scenario is good for global, unexpected but moderately known issues while VBS error handling is good to prevent anything unexpected, reportable and help to maintain the flow of test.

Let's recap the **MVC model** as of now. We can develop presentation layer (View) by n-number of means. This layer will speak the language of user. This layer can comprise any user requirement for example test suite, path, location of results, ALM(or known as QC) or file system library or script

path, scheduler information and association of recovery or any other resources. The controller (i.e. driver) can be written in any language (Preferred VBScript due to UFT language) and it understands the input provided by presentation layer. It then interacts with our core framework design (i.e. Model) to run the test. Model (i.e. design) is the crucial part. This can be done by n-number of ways and depends on the level of granularity we require to reuse our components. It also relies heavily on the type of development model. If our requirement frequently changes then it is advisable to use finer granularity to make the global changes at one place without compromising the scalability and flexibility.



So, in nutshell, our framework does not bound with any one language, tool, application or fixed architecture but it is an evolving process to improve every bit and pieces in the complete automation architecture

STEP6: Let's finish it with valuable ornaments – End to end solution

We have seen the core component of automation framework but it is very raw as of now. We need to wrap this model more humane. What it means that if we can incorporate more user friendly facilities to our framework

- Scheduler to start automation at pre-define time and without manual intervention (e.g. nightly run)
- Create a report of crucial application data e.g. User data, order information etc.
- Create a report of Summarize test run e.g. tests pass, tests fail, and time taken by run
- Create a report of error with screenshot e.g. environment error, application error, UFT error.
- Email the report to all stakeholder after run
- Run test on multiple machine
- Pre-Automation clean up scripts
- Post-Automation Clean up scripts

