TEST AUTOMATION EFFORT ESTIMATION

- Lesson Learnt & Recommendations

Babu Narayanan

1. Candidates for test automation.

One of the classical mistakes of the test automation team is: 'NOT choosing right test cases for automation'.

For any smart customer, the test automation scripts are only a support device to manual testing, NOT to bump off the later. In that case, the customer will be more focused on the return on investment (ROI) for each of test automation script built (as the initial investment is high!). So choose the tangible test cases to automate against each phases of development (and demonstrate the same to the customer).

How to find good test case candidates?

Sl. No	*Test Case Complexity*	*Number of actions*	*Number of verifications*	*Good candidates (~ No of executions)*
1	Simple	< 5	< 5	> 15 executions
2	Medium	> 5 - < 15	> 5 - < 10	> 8 - 10 executions
3	Complex	> 15 - < 25	> 10 - < 15	> 5 - 8 executions

Please be aware that step comprises of actions and verification points (or expected results as some spell). Mostly actions are direct method or function calls to test tool scripting language but the verification points are not of that kind.

Why do you need phase-wise test automation?
Most of the test automation projects fail which is mostly due to application rapid change, unsuitable test cases, shaky frameworks and/or scripting issues. Also it summarizes that test automation projects catch fewer defects than it is supposed to do.
** Mostly budget overrun than estimated.

The root casual analysis showed us the necessity of phase-wise test automation than one-go test automation. So advice that test automation needs to kick off with critical test cases that are of good candidate type and then slowly branching out to other mediums as required. This solution helps the customer by lesser maintenance costs and more business for you.

Also remember that framework needs constant updates along with script development process and thereby it becomes harder to maintain the framework incase if you have many scripts to develop in parallel.

What test type to be automated?
It is always good to script 'integration and/or system' functional test cases as they mostly bundle component level test cases within them. This would reduce your effort further and find good defects (Yep! Of course, this is primary objective of any test automation project) too. Please note that if you have good framework, then you can change the test scope and/or test type using configuration files.

2. Factors that affects test automation estimation.

The following factors may have varying impact on the test automation effort calculation exercise.

Sl. No	*Type*	*Factor*	*Impact*	*Remarks*
1	Framework	Availability	High	Good framework makes your scripting, debugging and maintenance easier. Do understand that framework needs continuous updating across the script development.
2	Application	Repeat functionality	High	It is quite easier to automate incase the functionality repeats across the application (Recommend to use keyword driven in such cases, as you do not end up in writing more action/verification based methods). If NOT, then the effort of building libraries and/or scripts is more linear in nature.
3	Project	Test Scope	High	If the complexity of application as well as the test scope is complex in nature, then it would consume huge efforts to automate each test case.
4	Test tool	Support to AUT	Medium	The test tool to be used may not support some application functionality and may cause overhead. You may find it more difficult to get started with open source scripting and/or tools.
5	Scripter	Skill	Medium	This costs project. The right skill packages of the scripter are very essential for any good test automation. If the customer NOT willing to provide the leverage on the estimate for this factor, do NOT forget to add the learning curve cost to the overall time.
6	Application	Custom Objects	Medium	The number of custom objects in the automation scope matters as it becomes overhead for the test automation team to built and maintain the libraries for them.
7	Application	Type (Web/ Client-Server / Mainframe)	Medium	For web application, any commercial test tool has amazing utilities and support. Otherwise, there are good possibilities that you need to spend huge effort in building libraries.
8	Application	Developed Language & Medium	Low	It matters as selected test tool does not support specific verification checkpoints.

3. Grouping steps to determine complexity.

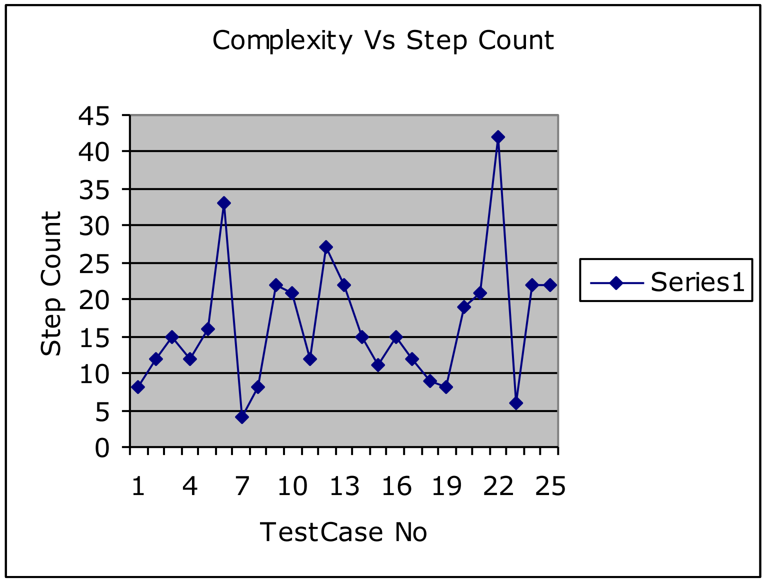
This is an important exercise as it may draw wrong overall effort in spite of depth application analysis.

STEP 1:

Suggest finding number of actions and verifications points for each test case (that are in automation scope) and then draw a chart to find the average actions, verification points and then the control limits for them. So that the complexity derivation will be based on the AUT not based on the generic industry standards.

Example,

TC	01	02	03	04	05	06	07	08	09	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25
STEPS	8	12	15	12	16	30	4	8	22	21	12	27	22	15	11	15	12	9	8	19	21	42	6	22	22



STEP 2:
Based on the data chart,

Average step count	= 16
Lower control limit	= 08
Upper control limit	= 25

So the complexity can be grouped as:-

Simple	≤	7 steps		
Medium	≥	8 steps	--	≤ 16 steps
Complex	≥	17 steps	--	≤ 25 steps

Recommendations:

1. Neither group test case steps too close, nor wide for labeling the complexity. Be aware that the pre-script development effort for each test script is considerable as the following activities are time-consuming operations:-
   1) Executing test case manually before scripting for confirming the successful operation.
   2) Test data selection and/or generation for the script.
   3) Script template creation (like header information, comments for steps, identifying the right reusable to be used from the repository and so on.)

   These efforts are highly based on the number of steps in the test case. Note that if test case varies by fewer steps, then this effort does not deviate much but incase it varies by many steps even this effort widely differs.

2. Also another factor in determining the complexity is the functionality repetition. If the test case is Complex by steps but the functionality is same as the other test case then this can be labeled as 'Medium or Simple' (based on the judgment).
3. If the test case steps count are more than upper control limit (~ 25 in this case) value then those additional steps need to be considered as another test case. For example, the TC - 06 containing 30 steps shall be labeled as
'1 complex + 1 simple (30-25)' test cases.

If the test case is marked as 'Complex' instead of 'Medium', understand that your efforts shoot up and hurts your customer. On other way of miscalculation, it hurts us. There by, this 'complexity grouping' is more of logical workout with data as input.

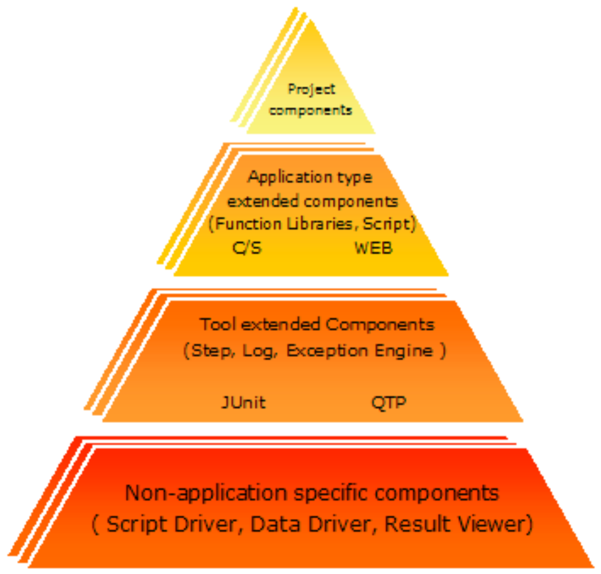
4. Framework design & estimation.

"We have experienced a significant increase in software reusability and an overall improvement in software quality due to the application programming concepts in the development and (re)use of semi finished software architectures rather than just single components and the bond between them, that is, their interaction." **- Wolfgang Pree [Pree94]**

There are many frameworks that are available commercially & as open-source which are specific to a test tool or wide-open. Also you find homebrew test automation frameworks too specific to test tools. These frameworks saves a lot of scripting, debugging and maintenance efforts but aware that the customization of framework (based on the application) are very essential.

Characteristics of any framework:

- Portable, extendable and reusable across and within projects.
- Ease of functionality Plug-ins/outs based on application version changes.
- Loosely coupled with the test tool wherever possible.
- Extended recovery system and exception handling to capture the unhandled errors and to run smoothly.
- Step, Log and Error Information provide easier debugging and customized reporting facilities of scripts.
- Ease of test data driven to the scripts and they need to be loosely coupled.
- Easily controllable and configurable test scope for every test run.
- Simple and easy integration of test automation components with test management, defect tracking and configuration management tools.


Project components
Application type extended components (Function Libraries, Script)
C/S
WEB
Tool extended Components (Step, Log, Exception Engine)
JUnit
QTP
Non-application specific components (Script Driver, Data Driver, Result Viewer)

Please note that these efforts have wide range as the framework size and scope purely depends on application nature, size and complexity.

It is always a good practice to create and/or customize the framework for initial needs and then add/ update components/ features and fine tune them as project goes.
Be aware that wrong framework choice may cost your project.

5. Scripting Effort Estimation

SL.NO	SUB COMPONENT	ESTIMATED EFFORT			REMARKS
		Simple (<8 steps)	Medium (8-16 steps)	Complex (17-25 steps)	
1	**Pre–Script Development**				
a	Test Case execution (Manual)				For 1 iteration (assuming scripter knows navigation)
b	Test data selection				For one data set (valid/invalid/erratic kind)
c	Script Template creation				Can use script template generation utility to avoid this.
d	Identify the required reusable				Assuming proper reusable traceability matrix presence.
2	**Script Development**				
a	Application map creation				Assuming the no of objects = number of actions
b	Base scripting				Normally all these go hand-in-hand. Separated for analysis & reasoning.
c	Add error/exception handling				
d	Implement framework elements				
3	**Script Execution**				
a	Script execution				For n iterations (~ average iteration count)
b	Verification & Reporting				Assuming there will minimal defect reporting.
	Total Effort per script				

<u>Keyword driven</u>
This total effort would vary if you choose key-word driven methodology but at the same time, the effort of building framework will be high (for initial design and scripting).

Do not use keyword driven approach for small projects.
These efforts may differ based on the above discussed (section 2) factors. Suggest you to perform PoC for 2 scripts from each class to confirm.
The negative test cases normally consume additional script efforts as the pattern changes.

Overall effort calculation may have the following components:-

1. Test Requirement gathering & Analysis
2. Framework design and development
3. Test Case development (incase the available manual test cases not compatible)
4. Script Development
5. Integration Testing and Baseline.
6. Test Management.

All these components shall include review (1/2 cycles).