International Journal of Innovative Research in Computer Science & Technology (IJIRCST)
ISSN: 2347-5552, Volume-2, Issue-4, July- 2014

Impact of Test Automation and Test Case Design Techniques - Challenges

Prof. Neelima V. Padmawar, Prof. Deepika A. Sarwate

***Abstract*-Test Automation is more than a set of tests run to generate apparent results. It includes designing testware, implementing automated test cases, and monitoring and interpreting a broad range of results. Automation by simply running test cases without human interaction doesn't provide interesting test exercises. This paper describes key elements of automated testing that need to be considered, models for testing that can be used for designing test automation architecture.**

The paper first develops a general framework for discussion of software testing and test automation. This includes a definition of test automation, a model for software tests, and a discussion of test oracles.

The remainder of the paper focuses on using various designing techniques for framework to plan for designing test cases.

Keywords- Test cases, Test Oracles, SUT, Decision Table

I. INTRODUCTION

Test Automation is more than a set of tests run to generate apparent results. It includes designing testware, implementing automated test cases, and monitoring and interpreting a broad range of results. Automation by simply running test cases without human interaction doesn't provide interesting test exercises.

Automated Software Tests :

Manual testing can be described as a situation where a person initiates each test, interacts with it, and interprets, analyzes, and reports the results. Software testing is automated when there is a mechanism for tester-free running of test cases.

Decision Tables :

Equivalence partitioning and boundary value analysis are very useful techniques. They are especially useful when testing input field validation at the user interface. However, lots of testing that we do as test analysts involves testing the business logic that sits underneath the user interface.[4]

Test Cases :

A test case in software engineering is a set of conditions or variables under which a tester will determine whether an application or software system is working correctly or not. The mechanism for determining whether a software program or system has passed or failed such a test is known as a test oracle [4]

II. CHALLENGES IN AUTOMATED TESTING

- To automate capture and comparison the data has to be machine readable, which is somewhat difficult for many classes of results including GUI displays, screen navigation, and program states.
- Results capture can be a huge task when you realize the world of possible results from running software. The actual results of running a test are much more than viewing information on the screen.
- Results interpretation is a second area that can spell trouble for an automation effort. We need to identify those things we are going to verify as 'the results' and we need to have an oracle to tell us what those results should be.1,2

III. KEY FACTORS IN AUTOMATED TESTING

- The first step in planning for test automation is to identify and understand some key factors about the SUT: Identify what software is to be tested, its specific components and features we want to test, and the environment surrounding the SUT. These factors are critical to the automation architecture. Additionally, understand the existing and available testware elements and tools for testing and test automation in the SUT's environment.
- After the SUT elements are identified the environment and interfaces must be considered. For large or complex SUTs there are often multiple environments to consider and the core programs to be tested may have several GUI and API interfaces. The immediate technical environment is important to automation because it provides the facilities and constraints on the most practical approaches.
- The form of the data for input and results capture is also important. Inputs and results may be keystrokes, data communication messages, pictures, sounds, digital device inputs or outputs, display information, etc. [1]

IV. SOFTWARE TEST ORACLES

Test oracles are the mechanisms for generation of expected results so we have something to compare with

Manuscript received July 19, 2014
Prof. Neelima V. Padmawar, JSPM Narhe Technical Campus-RSSCA,Pune, India (e-mail- padmawarneelima@gmail.com)
Prof. Deepika A. Sarwate, JSPM Narhe Technical Campus-RSSCA,Pune, India (e-mail- herminder.singh13@gmail.com)

the SUT responses. For manual testing the oracle is most often the human running the tests. With automation of tests it becomes necessary to automatically generate the expected results in a computer compatible form, and then have the computer system compare actual with expected results.

A tester can interpret system behaviours such as GUI navigation and can also perform diverse functions like arithmetic computations, string concatenation, or data base interrogations. Humans know that many factors such as dates and times change and can factor that in when generating or comparing results. The human oracle is so capable that they are often unaware that they are continually modelling program behaviour and comparing results.[2]

V. MODEL OF TESTING

Software testing involves more than feeding inputs to a program and observing results. Software today also has states and interacts with stored data and the computer environment.

Given the model, we might split the testing problem into several smaller problems. The majority of functions performed by a program might be tested through a public or private API. Automating tests run through a programming interface is usually easier and more reliable than through a GUI. A GUI front-end can then be tested independently of the program functions. This means that functional testing is done with programs that can directly feed and retrieve values through the API. GUI testing becomes an exercise to see that values typed into fields are fed into the back-end correctly, that values passed from the back-end are displayed correctly, and display windows are traversed based on the rules for GUI navigation. Figure 1 illustrates a splitting of the testing problem for the SUT.[3]

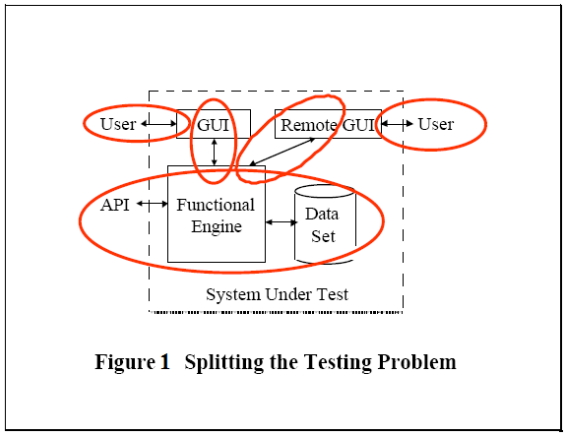

Figure 1 Splitting the Testing Problem

VI. AUTOMATED TEST ARCHITECTURE DESCRIPTIONS

We have now collected the information needed to describe the architecture. This is the step in the process where "a miracle occurs." Alternatives are weighed based on the requirements and available information. Factors such as the availability and accessibility of inputs and results, the availability or ease of creation of tools, stability of the SUT, availability and practicality of oracles, testing priorities, and resource availability must be weighed with the technical requirements. The result should be practical, cost effective automation architecture.

There are two parts to the description of the automation architecture. The first is a structural description that shows the elements and connections between them. Like a data flow diagram, information can be displayed and its control and movement depicted. An example of a structure diagram is shown in Figure 2.

The second part of the description is the sequence or flow of events. What event starts what process, and when. This second part explains the process for the automated test sequence. A typical sequence is shown below. Not all steps need to be automated.

There is always a cost tradeoff to be considered. Each step must be done, but the cost of automating them may be prohibitive for some of the activities.[3]

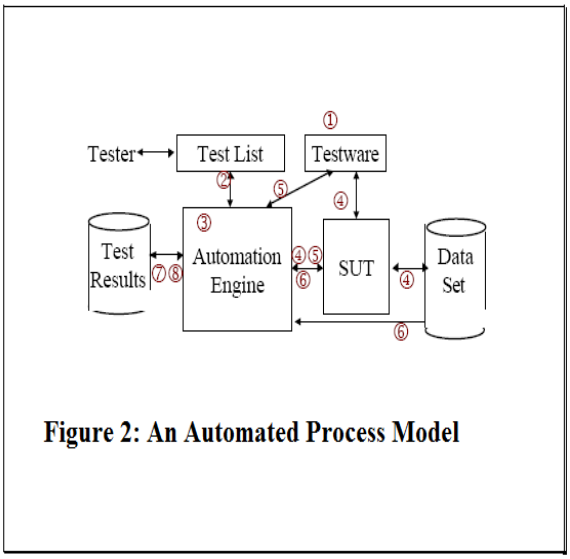

Figure 2: An Automated Process Model

Flow of events :

a) Test ware version control and configuration management
b) Selecting the subset of test cases to run
c) Set-up and/or record environmental variables
d) Run the test exercises
e) Monitor test activities
f) Capture relevant results
g) Compare actual with expected results
h) Report analysis of pass/fail

VII. DESIGN TECHNIQUES

A Decision Table is the method used to build a complete set of test cases without using the internal structure of the program in question. In order to create test cases we use a table to contain the input and output values of a program. Such a table is split up into four sections as shown below in fig 3.

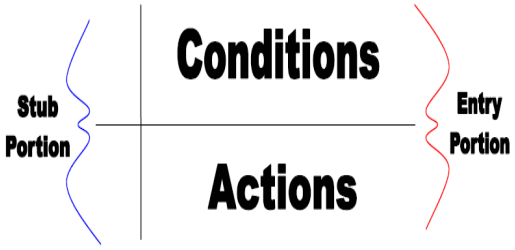

Figure 3 The Basic Structure of a Decision Table

In fig 3 there are two lines which divide the table into its main structure. The solid vertical line separates the Stub and Entry portions of the table, and the solid horizontal line is the boundary between the Conditions and Actions. So these lines separate the table into four portions, Condition Stub, Action Stub, Condition Entries and Action Entries.[5]

A column in the entry portion of the table is known as a rule. Values which are in the condition entry columns are known as inputs and values inside the action entry portions are known as outputs. Outputs are calculated depending on the inputs and specification of the program.

In fig 4 there is an example of a typical Decision Table. The inputs in this given table derive the outputs depending on what conditions these inputs meet. Notice the use of "-"in the table below, these are known as don't care entries. Don't care entries are normally viewed as being false values which don't require the value to define the output.[5]

Stub	*Rule 1*	*Rule 2*	*Rule 3*	*Rule 4*	*Rule 5*	*Rule 6*
c1	T	T	T	F	-	T
c2	F	T	T	T	-	T
c3	T	T	-	T	T	T
c4	T	F	F	T	T	T
a1	X	X		X	X	X
a2		X				
a3	X			X		
a4			X			X

Figure 4 a Typical Structure of a Decision Table

Figure 4 shows its values from the inputs as true (T) or false (F) values which are binary conditions, tables which use binary conditions are known as limited entry decision tables. Tables which use multiple conditions are known as extended entry decision tables. One important aspect to notice about decision tables is that they aren't imperative as that they don't apply any particular order over the conditions or actions.

VIII. CONCLUSION

Test automation is much more than computers launching test programs. Automation architecture includes many factors that need to be understood and addressed before automating testing. It begins with understanding test requirements, the SUT, and the test environment.

Using modelling and Design techniques like Decision Table, we can understand and analyze the testing and automation problems. This information can then be applied to describe a test automation environment using structural diagrams and event sequences.

IX. FUTURE SCOPE

But still event handling cannot be handling by using this automated architecture so for this we need to redesign model so that it can handle following.

1. GUI testing
2. Event Handling
3. Track of multiple screens

REFERENCES

[1] Douglas Hoffman, "A Taxonomy for Test Oracles," Proceedings of 11th International Quality Week, May, 1998
[2] Douglas Hoffman, "Heuristic Test Oracles," Software Test & Quality Engineering, V1, I2, March/April 1999
[3] Cem Kaner and Douglas Hoffman, "Thoughts on Oracles and Software Test Automation," Proceedings of 12th International Quality Week, May, 1999
[4] Advanced Software Testing: Volume1.
[5] Cai Ferriday 345399 Tutors: Dr. Roggenbach Prof. Schlingloff January 7th, 2007

Prof. Neelima V Padmawar has Masters of Computer Application. She has an experience of 8 years in teaching and has keen interest in Software Engineering, Testing and Object Oriented Analysis & Design. She has published several papers in National and International conferences and journals. She has authored 40+ text books for Pune, Mumbai and Gujarat Universities. She has been awarded as 'Best Citizen of Pusad, Dist:Yavatmal(MH)' in 2010.

Prof. Deepika A Sarwate has Masters of Computer Application. She has an experience of 12 years in teaching and has keen interest in Computer Networking, Operating Systems and Object Oriented Analysis & Design. She has published several papers in National and International conferences and journals.