International Journal of Information Technology and Management

Vol. VIII, Issue No. XII, May-2015, ISSN 2249-4510

APPROACH TO SELECT TEST CASE GENERATION IN SOFTWARE TESTING PROCESS FOR AUTOMATED TESTING AND DATA MINING

International Journal of Information Technology and Management
Vol. VIII, Issue No. XII, May-2015, ISSN 2249-4510

Approach to Select Test Case Generation in Software Testing Process for Automated Testing and Data Mining

Shiv Kumar[1] Dr. Rajan Anand Malik[2]

[1]JJTU Scholar, Jhunjhunu

[2]Director JEMTEC Greater Noida

Abstract – The design of an appropriate test suite for software testing is a challenging task. It requires a suitable tradeoff between effectiveness, e.g., a sufficient amount of test cases to satisfy the test goals of a given coverage criterion, and efficiency, e.g., a redundancy-reduced selection of test cases.

Keywords: Software Testing, Data Mining, Automation

- - - - - - - - - - - - - - X - - - - - - - - - - - - - -

INTRODUCTION

Data mining, *the extraction of hidden predictive information from large databases*, is a powerful new technology with great potential to help companies focus on the most important information in their data warehouses. Data mining tools predict future trends and behaviors, allowing businesses to make proactive, knowledge-driven decisions. The automated, prospective analyses offered by data mining move beyond the analyses of past events provided by retrospective tools typical of decision support systems. Data mining tools can answer business questions that traditionally were too time consuming to resolve. They scour databases for hidden patterns, finding predictive information that experts may miss because it lies outside their expectations.

REVIEW OF LITERATURE:

There has been a significant amount of work in automatic test case generation that attempts to increase the amount of observed behavior. Despite of these wide researches, there have been few efforts on representing an all-around classification, which covers all existing automatic test case generation approaches. A general classification for these techniques is presented in [5-7]. In [8] a classification framework for automatic test case generation methods is presented which is based on software development phase in which testing is applied. In [9] a classification of search-based automatic test case generation approaches is represented there are different reasons to automate test case generation task in software testing. Some of the most important reasons are as follows.

Reducing the cost of software testing - During testing phase the cost can increase more than the expected value due to inappropriate test cases. These inappropriate test cases cause wastage of organizational resources as well as time. There is a need to minimize the cost for getting an acceptable product [10]. Reducing human errors: In order to find out how a test case is valid there is no definite mechanism. It basically depends on the testers concerned of the requirement. In this process there are lots of human errors and tester basic skill level taken into consideration. This leads to the inclusion of bugs in the system after testing. To overcome this problem, automatic test case generation phase should be considered [11].

Test automation has always been an attractive alternative to expensive, time consuming and inconsistent manual testing. Key program factors include: the development paradigm, software quality objectives (1), and deployment velocity. When, how and how much test automation to apply against a program is dependent on these factors the return on investment must align with these factors; otherwise, the long-term success of the test automation effort will be in jeopardy and almost certainly fail (2). The most common key program factors are:

- Development paradigm (Agile, non-Agile, instrumented, non-instrumented)
- Quality objectives (defect escape velocity)
- Target deployment velocity (volume of new/enhanced functionality per release)

A test automation program should be considered when key program factors indicate the overall development program is not meeting expectations and there are no cost-effective alternatives to test automation (3, 4). The key indicators from a program objective would be:

- quality objectives are not being met
- defect escape velocity into production is deemed unacceptable
- target deployment velocity is not being met because testing is perceived as a bottleneck
- testing is not being completed within the assigned timeframe

It should be noted that testing is often not the “true” bottleneck, but as long as it is perceived as the bottleneck other systemic quality issues will not be addressed. Test automation is one way to remove the perception of a testing bottleneck empowering the testing team while allowing the program to focus on other systemic quality issues for example the initial quality of the code.

Selection of Correct Test Cases for Automation Testing:

Automation does not overpower or replaces manual testing but it compliments it. Like manual, automation too needs a strategy with proper planning, monitoring & control. Automation, when put into practiced correctly, can become an asset to the team, project and ultimately to the organization.

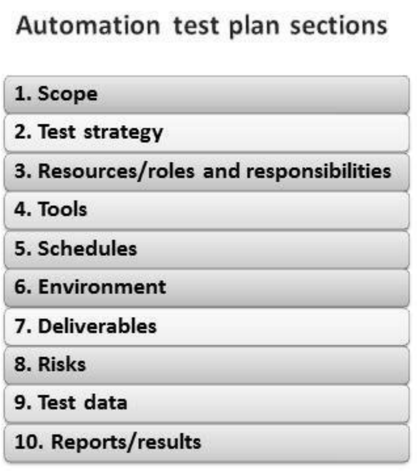
There are many advantages of automation; here are few important to mention:

- Useful to execute the routine tasks like smoke tests and regression tests.
- Useful in preparing the test data.
- Helps to execute the test cases which involve complex business logic.
- Good to execute the cross platform test cases (like different OS, browsers etc.)
- Great to execute the test cases which are a bit difficult to execute manually.
- When the number of iterations of the test case executions are not known.

Many a time stakeholders feel that test automation acts as a support tool for manual testing, so it’s vital to recognize that automation is the best way to increase the effectiveness, efficiency and coverage of testing. It not only saves time but also improves accuracy as repetitive tasks via manual approach can prone to human errors and can be time consuming.

For an automation project, it needs expenditure for:

1. Automation tools
2. Add –in for test management tool integration
3. Add-in to support AUT (like SAP, oracle etc)
4. Framework set up
5. Tool specific training



SCOPE

- Choose the test cases/scenarios that are to be regressed over and over across multiple cycles.
- Sometimes the simplest of test cases need lots of complicated solutions to be automated. If these are just for a one time use, it obviously does not make sense. Reusability should be our focus.
- Automation Testing does not/cannot perform exploratory testing.

TEST STRATEGY

- This section is referred to as Framework in the automation world. Some frameworks are extremely challenging to create and also are effective – but time, effort and competency wise they are demanding. Always look for a middle ground and do the best we can without jeopardizing over utilization of resources.
- Decide on coding best practices to be used, naming conventions, locations for test assets to be stored, format of test results , etc. to

maintain uniformity and increase productivity.

Orthogonal array approach for Test Case Optimization [12]:

The following steps are followed to construct the orthogonal array for testing a program with f factors, each factor having p levels:

1. Check if p is a prime number.

2. In case where each factor has different levels, check whether the highest level is a prime number.

3. If the highest level not a prime, identify the next highest prime number.

4. Check if f <=p+1.If not, check if f is a prime number, else identify the next highest prime number.

5. There exists an OA (Orthogonal Array) with p2 rows and (p+1) columns.

6. When p=3, we have an OA with 9 rows and 4 columns.

7. We construct "p tuples" (e1, e2, ….ep) as follows:

e1 = (0, 1, 2,….,p-1) = (0,1,2)

e2 = (1, 2,……,p) = (1,2,3)

ei=(ei-1 + e1) mod p, for i= 3 to p

e3 = (e2 + e1) mod 3 = (1, 3, 2)

CONCLUSION:

In this paper we found that a series of software programs to validate test output against specified test conditions. It's the best way of executing repetitive test cases using some software/tool which controls the test execution. There are various different factors that determine the effectiveness of the testing. To keep the client happy, the managers have to constantly look for opportunities to decrease the overall cost of testing. Matured organizations are now looking at newer and long term solutions for defining the test effectiveness of their testing functions.

Test automation is getting rid of repetitive manual tests and replacing those with systematic programs using automation tools.

REFERENCES:

1. The Automated Testing Handbook by Linda G. Hayes

2. http://searchsoftwarequality.techtarget.com/tip/Test Automation-When-how-and-how-much

3. http://www.tutorialspoint.com/software_testing/software_testing_overview.htm

4. http://www.righthandtech.com/software - testing.php

5. R. Jeevarathinam and A. S. Thanamani, "Towards Test Cases Generation from Software Specifications", International Journal of Engineering Science and Technology, vol. 2, Issue 11, (2010), pp. 6578-6584.

6. A. Arcuri and X. Yao, "Search based software testing of object-oriented containers", Information Sciences, vol. 178, no. 15, (2008) August, pp. 3075-3095.

7. E. Alba and F. Chicano, "Observations in using Parallel and Sequential Evolutionary Algorithms for Automatic Software Testing", Computers & Operations Research, vol. 35, no. 10, (2008) October, pp. 3161– 3183.

8. M. Prasanna, S. N. Sivanandam, R. Venkatesan, R. Sundarrajan, "A Survey on Automatic Test Case generation", Academic Open Internet Journal, vol. 15, (2005).

9. P. McMinn, "Search-based software test data generation: A survey", Software Testing, Verification & Reliability, vol. 14, no. 2, (2004) June, pp. 105–156.

10. A. Sharma, A. Jadhav, P. R. Srivastava and R. Goyal, "Test cost optimization using tabu search", J. Soft. Eng. Appl., vol. 3, no. 5, (2010), pp. 477–486.

11. V. Rajappa, A. Biradar, S. Panda, "Efficient software test case generation using genetic algorithm based graph theory", Proceedings of the First International Conference on Emerging Trends in Engineering and Technology, (2008), pp. 298-303.

12. Shubhra Banerji , Orthogonal Array Approach for Test Case Optimization, International Journal of Advanced Research in Computer and Communication Engineering Vol. 1, Issue 9, November 2012, online available at: http://www.ijarcce.com/upload/november/5-

Orthogonal%20Array%20Approach%20for%2
0Test.pdf