Automated Test Oracle Methods

Rupinder Kaur
Department of IT
UIET,Panjab University,
Chandigarh,India

***Abstract*—Software testing is used to improve software quality by finding fault in the software. To verify the software behavior, testers require test oracle, which is a reliable source of expected software behavior and provides outputs for any input specified in the software specifications and a comparator to verify actual results generated by the software. While test automation requires automated oracle support, oracle automation is considered as a challenging task. These challenges include the automation required in expected output generation and results verification. This paper presents the overview of test oracle and the methods available for the same.**

Keywords - Software Engineering; Software Testing, Automated Software Testing; Automated Test Oracle, Expected Output Generation.

I. INTRODUCTION

To improve software quality and reliability, software testing has been used as a process of finding errors and failures in software products. Because of the cost of software testing process in terms of time, financial resources, complete testing is practically impossible. Test Automation is one of the main approaches that have been applied to decrease the testing costs. While test automation requires automated test execution and results verification, which is called Automated Test Oracle, oracle automation is considered to support test automation framework. A test oracle is a reliable source of expected outputs. It has been applied to verify test case results that executed on Software Under Test (SUT). A Complete Test Oracle is the one that provide correct outputs for any input specified in software specifications. Since the available manual oracles are costly and unreliable, automated and complete test oracles will be required to ensure the testing quality while reducing the testing costs. This paper explains the process of using an automated test oracle and some methods for automated testing.

II. TEST ORACLE PROCESS

In testing literature, software's generated outputs that need to be evaluated are called *actual outputs*, and the correct outputs that are used to evaluate actual outputs are called *expected outputs* [9]. Therefore, An oracle is a complete and reliable source of expected outputs and a tool to find faults. Simply this tool makes a comparison between actual and expected outputs. The process of finding correct and reliable

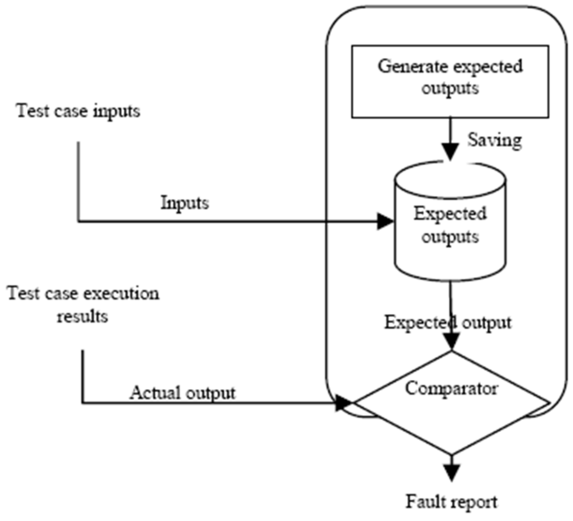

Figure 1. Test Oracle Process

expected outputs is called *oracle problem*[1] .According to [2], *oracle information* and *oracle procedure* are building blocks of a test oracle. The former is a source of expected outputs and the last is the comparator. Any modification in oracle information and using different comparator may obtain in different oracles. Possible test oracle process activities are:

1) Expected outputs generation.
2) Saving the generated outputs

3) Execute the test cases
4) Compare expected and actual outputs
5) Decide if there is a fault or not
Figure 1 depicts the oracle process and its activities. Note that test case execution is not part of test oracle, but it is part of the oracle process.

III.OVERVIEW OF THE VERIFICATION PROCESS

The Verification concept suggests separating the verification action from the actual test cases by forming a separate verification item. It
contains data items and a verification engine, as well as means to present the results (Fig. 2).

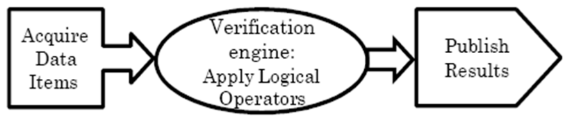

Figure 2 Verification Structure

Verification item definition; A verification item is an external investigator entity that anticipates, observes, controls, signals and documents selective states and occurrences represented by data items during and at the end of the TC execution [3,19]. They can be used to determine the exact location and time of the occurrence of a fault. When applied to the TC final output, the verification process is a logical representation of the test oracle.
Verification item values; A verification item is composed of input data items, arithmetic and logical operators to be executed on the input data to determine its correctness, and a result distribution method, as illustrated in Fig. 1.
Verification Item Requirements; In light of the growing complexity of modern applications and the data units involved, Verification items should be able to access all types of data and storage. For example, data can be represented by simultaneously using relational databases, XML, encrypted, compressed and coded files, as well as dynamic data locations [3]. Verification items are stored, maintained, managed, controlled and operated as external entities separated from the actual TC data, preferably as API calls .A Verification item should interface atleast one reporting system, to facilitate the result distribution mechanism. It should additionally be additionally be fast and efficient as verification actions may frequently be required during the TC execution thus should not hinder the execution performance. Finally, a Verification item should be able to handle complex logic structures, since the actual verification may be the result of relatively complex logic and multiple accesses to the data items, including performing complex calculations. Being an independent identity, a Verification can be reused to determine the correctness of occurrences in various TC's. Figure below explains the cross interaction of test cases possibly providing common verification .It in fact makes sense that TCs with similar functionality will share common verification.

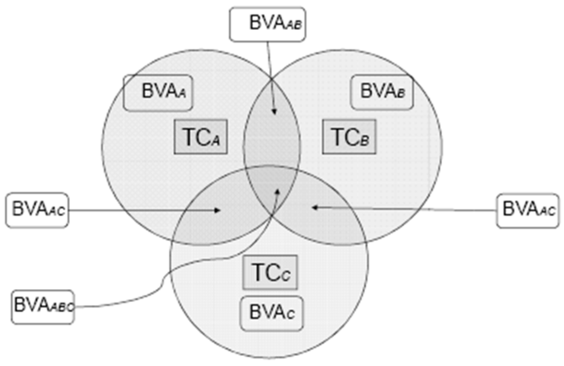

Figure 3.Verification Process reuse
Verification data as external entities to the actual test cases: Having BVA external to a specific TC will enable storage of verification items using any structures or architecture. This concept resembles the Dynamic Fault Trees (DFT) structure [4], yet here the hierarchy represents metadata structure reflecting the actual attributes at each level may differ accordingly to their use, so the upper level will hold attributes such as environment annotation, data storage naming and accessibility path etc. A lower level may for example, specify

items place holders, or symbols for logical operators to be applied during the Verification process. The separation of the verification mechanism from the SUT may also encourage reuse of the action in a semi manual pattern. Thus a manual tester would be able to use the verification action to perform a sophisticated verification that could otherwise have been hardly achieved, such as access an encryption and zipped data and validate its content without investing excessive efforts in learning the data structure .A practical example can be a verification tree for checking the validity of a personal address as represented in a customer Relationship Management (CRM), Billing or Management systems.

Verification Process and Test Oracle

The Verification process can then be applied on the TC output and results. This action is required because determining the correct outputs of tested application presents a real challenge for many reasons, for example because it can be simultaneously represented in various data forms. Thus the TC outcome is in fact the result of the operation of a vector of verification items on the application output, forming the logical representation of this test oracle .the Verification mechanism, however, applies to specific cases

where the test oracle can be represented by the Verification engine, for example, true oracles[4].We thus acknowledge that not all test oracles can be operated by this method, yet trust that a substantial amount of oracles can.

IV. TEST ORACLE AND EXPECTED OUTPUT GENERATION AUTOMATIC METHODS

Automatic test oracle can apply to overcome the oracle challenges and provide a complete test oracle. The process of using an automated oracle in software testing is shown in Figure 2 [5].

Automated oracle requires a simulated model of SUT. In order to provide a reliable oracle, it is suggested that the simulated model behave like the software under test and automatically generate correct expected outputs for every possible inputs specified in the software documentations. There are a few researches [8] on automated oracle. The following subsections explain them in detail.

A: N-Version Diverse Systems and M-Model Program Testing

It can be implemented using various automation tools.

Automated Oracle Activities-

- Expected output generation
- Saving the generated outputs
- Searching for generated outputs
- Comparison

Limitations-

- Requires various implementations of system functionalities
- High cost
- Could not test flow of events
- Still not reliable

B: Decision Table

It is implemented using a table [6].

Automated Oracle Activities-

- Saving the generated outputs
- Searching for generated outputs (if database uses to save the inputs and outputs)

Limitations-

- Manual output generation
- No automated comparator
- Only provides an structured approach for saving and indexing the I/O pairs

C: IFN Regression Tester

This method is implemented using IFN.

Automated Oracle Activities-

- Expected output generation
- Saving the generated outputs
- Searching for generated outputs
- Comparison

Limitations-

- Only applicable in regression testing

- Requires a reliable legacy system
- Requires additional knowledge for IFN modeling
- Could not test flow of events

D: AI Planner Test Oracle

The automation tool used here is AI planning.

Automated Oracle Activities-

- Expected output generation
- Saving the generated outputs
- Searching for generated outputs
- Comparison

Limitations

- Based on GUI state comparison
- Only applicable in GUI testing
- Requires formal model of the GUI to be tested
- Requires reliable documentations to provide trusted formal model
- High cost

E : ANN Based Test Oracle

This method is implemented using ANN [5].

Automated Oracle Activities-

- Expected output generation(only in regression testing)
- Saving the generated outputs
- Searching for expected outputs
- Comparison

Limitations

- Manual expected output generation (will be
- automated in regression testing)
- Requires additional knowledge
- Reliable in only data-centric applications
- Could not test flow of events
- Still not reliable in non-deterministic applications

V. DISCUSSION AND CONCLUSION

In this work, the concept of verification process is discussed, which is an external observer to the outcomes and results, intermediate and final occurring during a TC execution. Thus it does not interfere in the working of SUT as it is developed, stored and maintained externally. It can also be reused. Then we propose a few methods for the automation of test oracle along with their advantages ,disadvantages and limitations.

There are several benefits of applying this verification mechanism. First, the hierarchical metadata trees can be reused on different test cases. Second, these structures can be used as external data repositories to the real world. Third, the same verification mechanism can be used for additional activities. Fourth, the maintainability of the TCs increases as updating of as specific field only requires change in the specific rule.

The disadvantages of this system include the tedious effort required for verification process development and trees population. The thorough familiarity with the applications' business rules and data structures required. It also calls for an intensive involvement of software architects and developers during the creation of the tree. Also, in this paper we can conclude that all of the proposed automated oracles have limitations.It seems that there is still no unique approach to completely automate all oracle activities in any environment. Therefore it is recommended that further research be undertaken to develop a comprehensive automated test oracle which applicable in any type of software testing while automate all oracle
activities.

REFERENCES

[1]Ammann, P., and Offutt, J.: "Introduction To Software Testing" (Camberidge University Press, 2008, 1th edn. 2008)

[2] Xie, Q., and Memon, A.M.: "Designing and comparing automated test oracles for GUI-based software applications", ACM Transactions on Software Engineering and Methodology, 2007, 16, (1), pp. 4

[3] D. Hoffman, "Using Test Oracles in Automation", in Spring 2003 Software Test Automation Conference. 2003, Software Quality Methods,LLC.
[4] D. Almog. "Verification Points for Better Testing Efficiency", in StarEast 2007. Orlanodo FA: SQE.
[5]K. Sullivan, J. Yang, D. Coppit, S. Khurshid, and D. Jackson. "Software assurance by bounded exhaustive testing". 2004: ACM New York, NY, USA.
[6] Vanmali, M., Last, M., and Kandel, A.: "Using a neural network in the software testing process", International Journal of Intelligent Systems, 2002, 17, (1), pp. 45-62
[7] Di Lucca, G.A., Fasolino, A.R., Faralli, F., and De Carlini, U.A.D.C.U.: "Testing Web applications", in Editor (Ed.)^(Eds.): 'Book Testing Web applications' (2002, edn.), pp. 310-319
[8]Seyed Reza Shahamiri, Wan Mohd Nasir Wan Kadir, Siti Zaiton : **"**A Comparative Study on Automated Software Test Oracle Methods", Fourth International Conference on Software Engineering Advances,2009
[9] Dani Almog, Tsipi Heart.: "Developing the Basic Verification Action (BVA) Structure Towards Test Oracle Automation".

Rupinder Kaur received a B.Tech degree in information technology from Baba Banda Singh Bahadur Engineering college, Fatehgarh Sahib,India. She received an M.Tech in IT from University Institute of Engg. And Technology, Panjab University, Chandigarh, India. She's now working as Asst. Professor in Lovely Professional University,India.