Software Testing In Research and Actual

Deepak Jain[#1], Praveen Bhatt [*2]

[#] *Research Scholar, Department of Computer Science, JJT University*
Jhunjhunun, Rajasthan India
[1]deypak@gmail.com

[*]*Faculty of Allied Science, SGI*
Samalkha, Panipat, India
[2]Praveen34592@rediffmail.com

Abstract-**This paper will provide a compressive view of Software Testing. The main aim of this paper is to put all the pertinent issues of a cohesivecontext. In the vastness of the field, each topic approaches and problems will only briefly tackled with the proper reference .there is no entire survey of software testing , but my intend to show here that , how theoretical and technical problems are challenge for a software tester and there are the large gap between practice and the state of art.**

Keywords— **Software Testing, Software Quality Assurance, Software Engineering.**

I. Introduction

Testing by nature can never conclude anything mathematically valid as its amount to taking a simple and trying to infer a generally valid judgment on the whole. To complicate things when the objects of testing includes software, in making the inference we cannot rely on any certain continuity property as in the testing of physical system. We can work towards making the sampling less and more systematic. We can try to incorporate quantities, measurable notions within the analysis of test results.

So we can say Testing is a challenging activity and can greatly contribute to the engineering of quality programs.

II. Software Testing

Definition1: Software testing consists of dynamic verification of the behaviour of the program on a finite Set of test cases, suitably selected from the usually Infinite execution domain against the specified expected behaviour[1].

Dynamic: Dynamic means to explicitly specify that testing implies executing the programs on valued inputs. To be precise, the input value alone is not always sufficient to determine a test, as system behaviour generally depends on the system state. Different from testing and complementary with it, are static analysis-techniques, such as peer review and inspection .formal verification techniques such as model checker, data-flow analyser. All these approaches are important, but the left outside the scope of this paper. I mainly focus here the testing the implementation.

Selected: Test criteria essentially differ in how they selected the test suite. Tester should be constantly aware the different techniques may yield largely different effect. , also depending on context. How to identify the most suitable selection criteria under given condition is very complex problem. In practice risk analysis techniques and in test engineering expertise are applied.

Expected: It must be possible to decide whether the observed outcomes of the program execution are acceptable or not. Otherwise the testing would be useless. The observed behaviour may be checked against user's expectation or against a specification.

III. Faults vs. Failure

Fault and Failure denotes with different notations. When a test condition is executed and the response is "fail", the means that the program exposed as undesired

[1]This reference is from a chapter within the guide to the software engineering body of knowledge, providing and overview.

Behaviour: properly this is called a failure. It can be defined as deviation of the delivered service from the function the program was intended for.

The originating cause of the failure is said a fault. A fault may remain dormant long time, until it is activated and bring the program to a state which, if propagated to the observables output, can lead to a failure: this immediate unstable state is indicated as error. Below chain expresses a causality relationship.

Fault→Error →Failure

IV. Test Selection

The problem of test case selection has been the largely dominating topic in software testing research to the extent that " software testing " is often take as a synonymous for " test case selection " . An important point is always keep in mind is that what makes a test a "good" one does not have a unique answer. But it change depending on context.

In general, a test criterion is a means of deciding which a "good" set of test cases should be.

V. Selection criteria based on Code

Code based testing has been dominating trend in software testing research during the late 70's and the 80's. One reason is certainly that in those years in which formal approaches to specification were much less mature and pursued than now. This criterion is known as path based. They map each test input to a unique path on the flow graph .the ideal and unreachable target of code- based testing would be the exhaustive coverage of all possible paths along the program control flow. The basic test hypothesis is that , by executing a path once , potential faults related to it will be revealed , i.e. it is assumed that every executing a same path will either fail or succeed.

Code –based testing is, in which a family of criteria is introduced, based on both control flow and data flow.

VI. Selection criteria based on specifications

In specification based testing, the code based derived in general form the documentation relative to program specifications. Depending on how these are expressed, largely different techniques are possible. Very early approaches was looked at input /output relation of the program "black-box" and manually derived equivalence classes , or the boundary value conditions , or the cause –effect graphs. Lots of researches have tried to automate the derivation of test cases from formal or semi formal specifications.

VII. Other Criteria

There are lots of other test criteria has been proposed, but the size limitation do not allow us to tackle them in details. The other criteria can be Error guessing, or mutation testing etc.

VIII. Selecting the Test cases is not only the Issue

There are other test related activities present technical and conceptual difficulties that are underrepresented in research like select tests, test outcome is acceptable or not, impact of failure and finding its direct cause.

These activities have received marginal attention in software testing research. One argument is that being these issues technological in kind, in contrast with the more theoretical and intuitive problem of test selection, the approach pursued are specific to an application context.

IX. Test Execution

As we discussed, if the code –based criteria is followed, it provides us with entry-exit paths over the graph model, the test input that execute the corresponding programs paths need to be found. If a specification based criteria relying on coverage of test case is adopted, then the test cases are paths over the corresponding to sequence of events, that are specified at the abstraction level of the specification. To derive concrete test cases, the labels of the specification language must be translated into corresponding labels at code level, and eventually into execution statements to be launched on the GUI of the used test tool. The testing task itself requires a large programming efforts to able to test a piece of a large system, that we need tosimulate the surrounding environment of the piece under tests. This is done by developing ad hoc drivers and stubs; some commercial test tool exists than can facilitate these tasks.

X. Test Oracles

The important component of testing is oracles. A test is meaning full only if it is possible to decide about its outcome.
An oracle is any (mechanical or human) agent that decides whether the program behaved correctly on a given test. The oracle is specified to the output a reject verdict if it observes a failure and approve otherwise. Different approaches can be taken .suppose in a scenario in which a limited number of test cases are executed, sometimes even derived manually; the oracle is tester himself/herself. Who can either inspect a posterior the test log, or even decide a priori, during test planning, the conditions that make a test successful and the code these conditions into the employed test driver.
In some cases, the oracle can be an earlier version of the system that we are going to replace with the one under test. A particular instance of this situation is regression testing. , in which the test outcome is compared with earlier version executions.. Generally speaking an oracle is derived from a specification of the expected behaviour.

XI. Analysis of Test Results

As I mentioned in Test cases selection that, the researchers continuously strive for finding "good" criteria. But what makes criteria better than other? Effectiveness must be associated with a test case or an entire suite, but beat effectiveness does not yield a universal interpretation. Some people misconceive the meaning of coverage measures and confuse coverage with effectiveness. We have already mentioned that one intuitive and diffuse practice is to count the number of failures or faults detected. The test criterion that found the highest number could be deemed the most useful. Even this measure has drawbacks; as tests are gathered and more and more faults are removed, what can we infer about the resulting quality of the tested program? For instance, if we continue testing and no new faults are found for a while what does this imply? That the program is "correct", or the test is ineffective?

XII. The notation of Software Reliability

Software reliability is the probability that the software will execute without the failure in a given environment for a given period of time. Particularly, to assess the software reliability "in a given environment", this input distribution should approximate as closely as possible the operational distribution for that environment. The extremesdefault of identifying an operational distribution for software system is one of the arguments brought by opponents of software reliability. However the practical approach proposed is define even a course operational profile by grouping different typologies of users and functionalities have demonstrated great effectiveness.

XIII. Keeping altogether in flawless process

There are several problems discussed so far , real big challenge ahead is to work out a unified process within which all these test tasks are gracefully complementing each other , and testing as a whole is not an activity detached from construction , but the two things , building and checking , become two face of same coin , two seamlessly integrated activities.

XIV. Test Phases

Testing of a large system is organised into phases, i.e. the testing task is portioned into a phased process, addressing at each step the testing of a subsystem. Integration testing refers to the testing of the interactions between subsystems along system composition. An incremental systematic approach should be taken, as opposed to a big-bang approach. The aim is to keep complexity under control and to eventually arrive at the final stage of the system with all the composing subsystem extensively tested. At each stage test selection is closely related with the object under test. Some white-box testing approaches proposed to derive integration test cases based on the call structure among modules and measure inter-procedural coverage. In object oriented system, integration test consist of interleaved sequence of module paths and massages, and they are derived considering the interaction pattern between objects, for instance the colorations or the client-server hierarchy.

XV. Test Patterns

Practical instruments to the design of complex systems are patterns. A design pattern is an abstract descriptionof a recurring problem. Together with a general arrangement of elements and procedures that has proved to be useful to solve it. Patterns are always been used by expert designers and engineers: they form their cultural expertise. Symmetric to design pattern comes the idea of identifying and logging interesting and recurrent patterns in the testing of complex systems. Unfortunately there is not much work in this sense. Certainly more research and empirical work towards the definition of the test patterns is desirable.

XVI. Summary

We think, we discussed the general overview regarding several complex facts .We discuss about the Test selection , Test phase s , Test pattern , Fault and failure. Etc. There is much room for automation in each of the involved activities; the tester's expertise remains essential as much as a need for approximate solution under constrained remains. Again Testing is a challenging and important activity. Let me conclude with the famous quotation from Knuth: Beware of bugs in the above code; I have only proved it correct, not tried it.

XVII. References

[1] Beth Gold-Bernstein and William Ruh. Enterprise Integration: The Essential Guide to Integration Solutions. Addison-Wesley, 2005.

[2] Collard, R. Test Design: Developing test cases from use cases. Software Testing & Quality Engineering Magazine, 1999. Vol. 1:4. pp. 30-37.

[3] D. Gelperin and B. Hetzel. The growth of software testing.Commun. ACM, 31(6):687{695, 1988.

[4] Edsger W. Dijkstra – Notes on Structured Programming

[5] E. Hiean;Rc-Mee;"Going Faster: Testing The Web Application", IEEE Software, Mar. 2002, pp. 60- 65

[6] FilippoRicca and Paolo Tonella, "Analysis and Testing of Web Applications", IEEE, 2001

[7] Glass, R. L. 1998. "Defining Quality Intuitively", IEEE Softw. 15, 3 (May. 1998), 103-104,107

[8] Goger S. Pressman, Software Engineering APractioner's Approach, McGraw-Hill, 2001

[9] Hong Zhu, Xudong He, "A Study of Integration Testing and Software Regression at the Integration Level", 0-7695-1372-7/01 2001 IEEE

[10] HenrikBaerbak Christensen – Systematic Testing should bot be a Topic in the Computer Science Curriculum!Hong Zhu, Xudong He, "A Study of Integration Testing and Software Regression at the Integration Level", 0-7695-1372-7/01 2001 IEEE
[11] James A. Whittaker – What Is Software Testing? And Why Is It So Hard?

[12] Jerry ZeyuGao et al, Testing and Quality Assurance for Component-Based Software, 2003

[13] John Watkins, Testing IT An Off-the-Shelf Software Testing Process, 2004

[14] Kim H. Veltman, "Syntactic and Semantic Interoperability: New Approaches to Knowledge and the Semantic Web", The New Review of Information Networking, vol. 7, 2001

[15] Learning, Proceedings of 22nd Conference on Software Engineering Education and Training, IEEE, pp 279-281.

[16] MagielBruntink – Testability of Object-Oriented Systems: a Metrics-based Approach

[17] Prowell, S. J. TML: a description language for Markov chain usage models. Information and Software Technology, 2000. Vol. 42:12. p. 825-833.

[18] Robert C. Martin – The Dependency Inversion Principle

[19] Ricca, F. "Analysis, testing and Re-structuring of Web applications" Proceedings of the 20th IEEE International Conference on Software Maintenance (ICSM'04), 2004 [27] Nasib S. Gill, "Factors affecting Effective Software Quality Management Revisitied",

[20] RituArora and Sanjay Goel (2009), Software Engineering approach for teaching development of Scalable Enterprise Applications, proceedings of 22nd Conference on Software Engineering Education and Training, IEEE, pp 105-112.

[21] Robinson, H. Finite state model-based testing on a shoestring. International Conference on Software Testing Analysis and Review, San Jose, California, USA, 1999.

[22] S. Robertson. An early start to testing: How to test requirements. Conference on Software Testing, 1996. Thimbleby, H. The directed Chinese Postman Problem. Software – Practice and Experience, 2003. Vol 33:11. pp. 1081-1096.