Global Journal of Computer Science and Technology: H Information & Technology
Volume 16 Issue 3 Version 1.0 Year 2016
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

Probability Testing a Component of Advance Software Testing

By Debashis Ghatak
West Bengal University of Technology

Abstract- The present invention relates to the Probability Testing which is a new testing technique under Software Testing. This invention belongs to Software Testing, especially a new intellectual testing technique. The Probability Testing introduces a new mathematical formula in Software Testing as well as Software Engineering. Probabilistic Testing Architecture which gives a brief idea of Probability Testing before the start of the software testing.

Keywords: *knowledge, thinking, assumption.*

GJCST-H Classification: *B.8.1 D.2.5 K.7.3*

Strictly as per the compliance and regulations of:

Probability Testing a Component of Advance Software Testing

Debashis Ghatak

Abstract- The present invention relates to the Probability Testing which is a new testing technique under Software Testing. This invention belongs to Software Testing, especially a new intellectual testing technique. The Probability Testing introduces a new mathematical formula in Software Testing as well as Software Engineering. Probabilistic Testing Architecture which gives a brief idea of Probability Testing before the start of the software testing.

Keywords: *knowledge, thinking, assumption.*

I. Introduction

Each and every incident in this world has done its own logic which can be fabricated by a set of mathematical rules. It has been noticed that technological people have some fabrication knowledge that software development section established by logic or follows the rules of mathematics whereas Software Testing section doesn't follow any logic or any mathematical rule. But this concept is truly wrong. Suppose a logic for calculating two amounts like x and y and get the result in z like "x+y=z". During the time of software development, a developer can assemble this logic with the help of "DATA STRUCTURE" and different programming language. Similarly, a "Software Tester" can check the entire development, according to the requirement as well as in a different way like "Mandatory Fragment, Mandatory Sub Fragment, Dependent Mandatory Sub Fragment, Different Condition, Boundary Value Analysis, Statement Coverage, Equivalent Partitioning and Exit Criteria". All the respective checking is directly or indirectly related to "Mathematics" or some "Mathematical Rule". The backbone of any Software stands on five unique sections such as "REQUIREMENT, LOGIC, DESIGN, CODE GENERATION, TESTING" (See Fig 1).

When a Software scamper in the real time scenario at that time it has been observed that the software has been failed due to a different unique situation rather than traditional scenario. Origination of Probability Testing is to avoid such kind of different unique situation which happens during the time of real time scenarioz. "Probability Testing" belong to two phases such as first "PLANNING" secondly "ACTING". "Software Tester" first prepares a mock unique situation that may be or must be happening during real time. Software tester will make a decision that which field is required to create such kind of scenario and activity (See Fig 2).

Author: *West Bengal University of Technology.*
e-mail: debashis.ghatak@hotmail.com

Human intelligence comes from inherent thinking power and way of thinking of a person. It varies from person to person wise. Planning is the blossom of THINKING and the ASSUMPTION power of the person. Before the preparation of any planning, it requires an extensive area of THINKING, ASSUMPTION and thoroughly KNOWLEDGE about that particular item. Probability Testing belongs to the "Real Time" unique situation oriented Testing. In the "Probability Testing", a Software Tester must be needed three things

1. KNOWLEDGE
2. THINKING
3. ASSUMPTION

Details Architectural Diagram of "Probability Testing" is as follows (See Fig 3).

In the "Probability Testing" corresponding "LOGIC" segment will prepare different mock real time "Logical Scenarios". "PLAN" is the consequence of "KNOWLEDGE + THINKING + ASSUMPTION". The mother of any intelligent logic comes from the methodically "KNOWLEDGE". Application of the "KNOWLEDGE" comes when the steam of "THINKING" starts on that "KNOWLEDGE". At that time it has been noticed that different types of "ASSUMPTION" will come in the mind automatically and then different types of "ASSUMPTION" will be converted into the different "PLAN" like "PLAN1,PLAN2,PLAN3,……,PLANn".

Any Software works on the basic functionality of some "Mandatory Fragment", "Mandatory Sub Fragment", "Dependent Mandatory Sub Fragment", "Different Conditional Fragment" and related non-functional Fragments. It is following "TOP DOWN" architecture of the "Data Structure" model.

The objective of "Probability Testing" is the good choice for "Software Testing". According to the derived formula for a total number of "Test Case" generation for "Software Testing", it has been noticed that "Software Testing" is a combination of the different level of Fragment in different in different ways. In terms of mathematics, it is called "Permutation". Any software is a combination of various "Mandatory Fragment", "Mandatory Sub Fragment", "Dependents Mandatory Sub Fragment", "Conditional Fragment" and "Nonfunctional Fragment". "Fragment" dependency can arise any part of any "Software" according to the requirement. Probability Testing follows the "Probability Theory" that is "Permutation" of n distinct objects has taken r at a

time such that nPr=n! / (n-r)! .Where some of the fragments are obtained as concealed in the test design.

During the time of "Probability Testing", the tester will generate different mock real time situation, according to the business requirement and business logic. According to the different mock probable scenario which may be happening in the real time on the basis of that Software Tester try to find out the different kind of defects on the basis of various scenarios for taking requisite corrective measures in the software coding and eliminating the defects.

After a release of the software globally there are a numerous number of bugs has been found in multiple specific situations. "Probability Testing" is very much required to avoid such kind of situation before releasing the software globally. "Probability Testing" is the place where testers have a freedom of thinking and implement the said, thinking capability during the time of real-time testing. Considering the following relation in the software testing,

$$Pi=\sum_{i=1}^{n} Mi+\sum_{l=1}^{n} CF\,l$$

Where CF= nCr

Let "M" belongs to the "Mandatory Fragment". "CF" which also belongs to the others "Child Fragment" which is correlated with "Mandatory Fragment" and chooses a different combination of respective "Child Fragment" corresponding "r" at a time. The above formula will help for the tester that in which way and how a Software Tester will prepare different mock real time scenario during the time of "Probability Testing" also it will track the total count of the mock real time scenario throughout the operational time.

Every different "Fragment" has a relationship as like as "Parent" with "Child" as "Mandatory Fragment" with "Child Fragment". During the time of "Probability Testing", each individual "Fragment" can be prepared by using the above formula.

Formation of mock real time situation, corresponding "Mandatory Fragment" should be treated as "Parent Fragment" related sub-fragment is treated as "Child Fragment". Creation of the Mock real time situation during the time of "Probability Testing" is basically a "Top down" approach and corresponding relational diagram will go into synchronously from Parent to Child (See Fig 4).

"Child Fragment" will come up according to the mock real time situation and in every different, unique case will diverge according to the different mock real time situation. During the time of "Probability Testing", a tester will generate mock real time scenario like

Any Unmanned Ground vehicle (UGV) or Unmanned Arial vehicle (UAV) directly controlled by satellite or some elevated power wireless communication system. The "Unmanned Ground vehicle (UGV) or Unmanned Arial vehicle (UAV)" will perform their assignments in the inaccessible area where human meddling is impossible. This is traditional system and traditional testing, which is sufficient for such kind of scenario. But "Probability Testing" is quite different, suppose if they said "Unmanned Ground vehicle (UGV) or Unmanned Arial vehicle (UAV)" moves under a long tunnel or hilly dense forest where satellite network or wireless communication system is not reached or not available such kind of situation, how the "Unmanned Ground vehicle (UGV) or Unmanned Arial vehicle (UAV)" will rescue and reinstate the communication with respective satellite or wireless communication system such kind of scenario based testing have done by Probability testing.

In the general insurance for a "Private Car Package" product, suppose a car owner has a private car like "Mercedes Benz". During the time of "Policy Generation" of the said new vehicle owner was taken the "Sum Insured" that is "$100000"along with that several "Risk Cover" was taken like "Theft", "Accessories" etc.. After a few days later person was filed an endorsement that is "Change in Sum Insured" Endorsement from "$100000" to "130000". After successfully passed the said endorsement the owner of the vehicle faced an accident where one person was dying and one person was severely injured. Both family members were filed different claims like "Death & Injury" against the policy of the receptive person. "General Insurance Company" was appointed a surveyor against the said claim to find out the authenticity of the respective claims. Suddenly the said owner of the vehicle drove on the hilly road and faced on the "Land Slide". The owner was spot death along with that vehicle was smashed.

In the above type of cases, the family members of the owner will file a "Death" as well as "Own Damage" claim against the respective policy number. Software Tester will test what types of defects have been generated such type of mock real time situation that has been created during the time of "Probability Testing".

"Probability Testing" will testing will test such kind of mock real time scenario oriented testing and find out the defects against that incident. In the above mock real time situation based testing is not possible either in "Integration" or "Regression Testing". Because "Integration" and "Regression Testing" have done on the basis of "Test Cases" and "Probability Testing" has done on the basis of "Mock real time Test Scenario". Knowledge, Assumption & Thinking ability are the key of "Probability Testing".

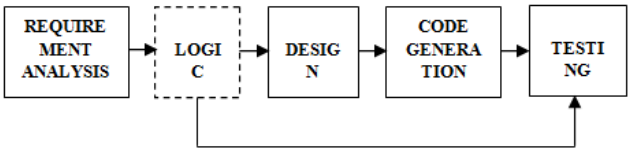

Figure 1: Enhance Software Design Architecture

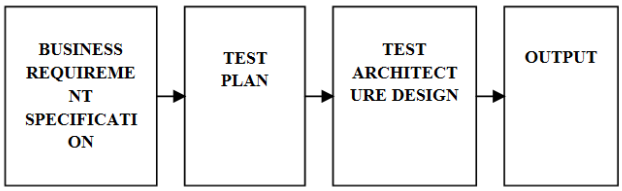

Figure 2: Basic Test Design Architecture

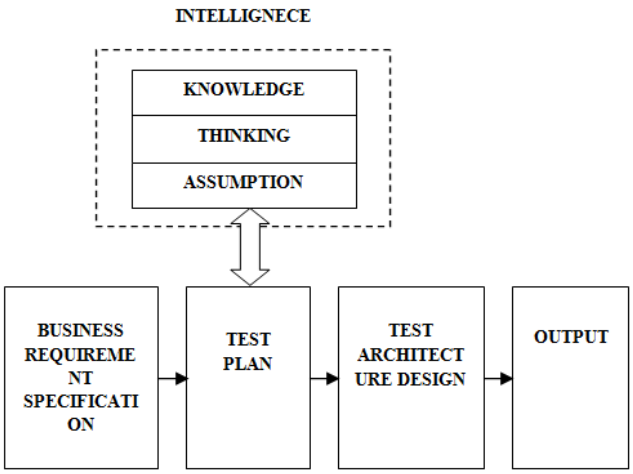

Figure 3: Probability Testing Architecture

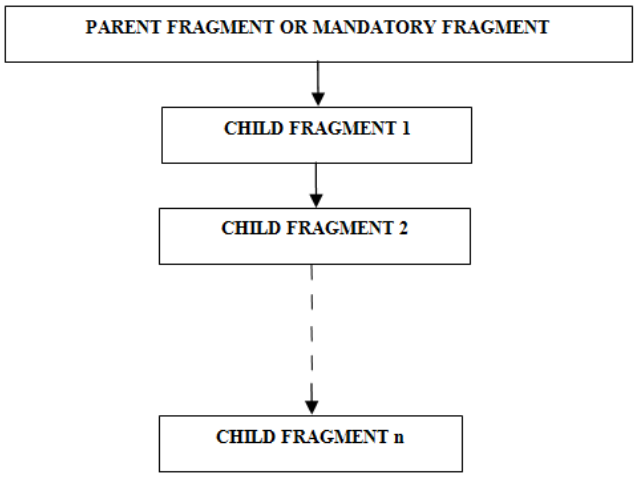

Figure 4: Probability Testing Fragment Relation

II. Conclusions

"Probability Testing" of Software Testing Which comprises, create mock real time intellectual situation, that the scenario which will probably happen in future during runtime and test the software on the basis of that scenario.

Said Software testing can be performed only after completion of "Integration Testing" and "Regression esting".

KNOWLEDGE, THINKING, ASSUMPTION is the key of Probability Testing.

III. Acknowledgment

The authors would like to thank Government of India for his/her moral support.

References Références Referencias

1. IEEE. 1993. IEEE Standard for Software Maintenance. IEEE Std 1219-1993. Institute of Electrical and Electronics Engineers, inc., NewYork, NY.
2. IEEE Standard for Software Verification and Validation Plans (Reaff.1992). IEEE Std 1012-1986.
3. IEEE Standard Glossary of Software Engineering Terminology. IEEE Std 610.12-1990.
4. JJ Kuhl, "Project Lifecycle Models: How They Differ and When to Use Them", 2002.
5. Sannella, M. J. 1994 Constraint Satisfaction Debugging for Interactive User Interfaces. Doctoral Thesis. UMI Order Number: UMI Order No. GAX95-09398., University of Washington.
6. Fundamental of Software Engineering: Rajib Mall
7. Software Engineering a Practitioner Approach: Roger S Pressman.

Global Journals Inc. (US) Guidelines Handbook 2016

www.GlobalJournals.org