SLOW INTELLIGENCE SYSTEM BASED SOFTWARE TESTING

Dr. T. S. Baskaran1, K. Suryakala2

1&2 Department of Computer Science, A.V.V.M. Sri Pushpam College, Poondi, India

Abstract: Testing of software is highly important than developing it, because doing activities without software is better than with erroneous software. Modern software world has plenty of software testing tools which can find errors only. Rectification is very difficult because it requires high level of human cleverness and computerization. Slow Intelligence System can provide intellect as well as mechanization in the area of error correction with the help of different phases such as enumeration, propagation, adaptation, elimination and concentration. This paper is mainly designed to make comfortable for both software developer and testers. So we include a unique concept called Slow Intelligence System which is used to find maximum possibilities of bug during development stage. It is implemented after the process of test cases. So it leads to detect exact possible errors in the software. So the developers can easily rectify their development consequences.

1. INTRODUCTION

Commonly computer software faces different problems such as erroneous calculation, inaccurate data edits, invalid matching & merging of data, incorrect result of data search, mistaken processing of data relationship, faulty coding and implementation of business rules, insufficient performance, confuse or misleading data, software usability by end users and out of date, defective results or performance, insufficient support of industry needs, poor interface with other systems and unacceptable file handling. In general testing has some common objectives execute a program with the target of finding an error, Check whether the system is in user expectations, Is it being the good breed of basic testing method.

To attain the mentioned objectives of tester must include:
• Find errors as early as possible and make sure they get fixed.
• Understand the application well.
• Study the functionality in detail to find where the bugs are likely to occur.
• Learn the code to ensure that each and every line of code is tested.
• Create test cases in such a way that testing is done to uncover the hidden bugs and also ensure that the software is usable and reliable.

A good testing program identifies the end of the “development” phase of the project, establishes the criteria for project acceptance, and establishes the start of the warranty period.

A tiny error in software application can cause server damage on critical application execution. Software testing is the process of finding errors by executing a program with sample inputs. [2]Choosing of sample inputs play a vital role to decide the trustiness of the testing tools while selecting a tool for a software testing the following various factors to be considered.

• Even though the testing not identifying any error in the software cannot be concluded that software is defect free.
• Same testing tool and same software cannot produce the same error sequence always.
• Accuracy depends on input to particular software.
• Error detection on software is completely differing with physical system.
• Detecting all of the different failure modes for software is generally infeasible.
• Discovering the design defects in software is very difficult.
All the possible values need to be tested and verified, but complete testing is infeasible.[3]

The remainder of this paper is organised as

Section 2.0 focuses about currently using testing methods related tools and the significance of SIS. Section 3.0 deals the proposed system methodology with performance measure between existing and proposed. Section 4.0 gives the concluding remarks.

2. EXISTING TESTING METHODS

In general software failure occurs because of logic, insufficient software testing, attitudinal changes among programmers, attacked by a hostile agent. Failure resulting from unanticipated applications or use. Software may also fail due to external causes such as Human error, Management laxity, Support systems, Cyber Security and Environment.

Software testing is a process of executing a program or application with the intent of finding the software bugs. [2]

Two ways of software testing is illustrated in fig.1

![Software testing ways](image)

Figure.1 Software testing ways

There are many different types of testing tools are available.[8] Each testing tool is capable of doing different things as each one has different abilities. Table 1 shows the different types of testing tools and its details.

<table>
<thead>
<tr>
<th>S. NO</th>
<th>TYPE OF TOOLS</th>
<th>EXAMPLE</th>
<th>DESCRIPTION</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Communication Testing Tool</td>
<td>Cheetah, Chariot, AdventNet Simulator</td>
<td>Used for communication purposes.</td>
</tr>
<tr>
<td>2</td>
<td>Coverage Tool</td>
<td>JavaCode Coverage, JFeature, NCover</td>
<td>Describe the degree to which the source code of a program is tested.</td>
</tr>
<tr>
<td>3</td>
<td>Database Testing Tool</td>
<td>MySQL Workbench, Webyog:SQLyog &amp; MONyog</td>
<td>Involves testing strategies such as quality control and quality assurance of the product databases</td>
</tr>
<tr>
<td>4</td>
<td>Functional Testing Tool</td>
<td>Marathon, Silk Test</td>
<td>Tests functionality of the whole system.</td>
</tr>
<tr>
<td>5</td>
<td>Integration Testing Tool</td>
<td>CruiseControl, Jenkins</td>
<td>Groups the modules in larger aggregates apply tests defined in an integration test plan to those aggregates.</td>
</tr>
<tr>
<td>6</td>
<td>Performance Testing Tool</td>
<td>AgileLoad, AppDynamics Pro</td>
<td>Determining the speed or effectiveness of a software.</td>
</tr>
<tr>
<td>7</td>
<td>Regression Testing Tool</td>
<td>Winrunner, QTP Regression Tester</td>
<td>Selection of already executed test cases which are re-executed to ensure existing functionalities work fine.</td>
</tr>
<tr>
<td>8</td>
<td>Requirement Management Tool</td>
<td>Analyst Pro, Doors, Caliber</td>
<td>Used for documenting.</td>
</tr>
<tr>
<td>9</td>
<td>Test Design Tool</td>
<td>Agile designer BenderRBT Coronys</td>
<td>Tools that help you decide what tests need to be executed. Test data and test case generators.</td>
</tr>
<tr>
<td>10</td>
<td>Unit Testing Tool</td>
<td>JUnit, NUnit, Jtest</td>
<td>The purpose is to validate that each unit of the software performs as designed. It is the smallest testable part of software.</td>
</tr>
<tr>
<td>11</td>
<td>Web Testing Tool</td>
<td>BugBuster, Weone</td>
<td>Complete testing of a web based system before going live can help address issues before the system is revealed to the public.</td>
</tr>
</tbody>
</table>

Some of the popularly using automation testing tools is HP Quick Test Professional, Selenium, IBM Rational Functional Tester, Silk Test, Test Complete, Testing Anywhere, Win Runner, Load Runner, Visual Studio Test Professional, WATIR, Star Team and HP Quality Center.

All the above mentioned tools can find only the mistakes of software but no effective tool is available to give suggestion to manage the software failures. Proposed SIS based method generates the suggestion and also to overcome the software issues.
SLOW INTELLIGENCE SYSTEM

Slow Intelligence Systems is a general purpose systems characterized by being able to improve performance over time through a process involving enumeration, propagation, adaptation, elimination and concentration [7]. It continuously learns, searches for new solutions, propagates the knowledge and shares its experience with other peers. A Slow Intelligence System differs from expert systems, in that the learning is not always obvious. [7] A Slow Intelligence System seems to be a slow learner because it carefully and gradually analyzes the environmental changes and absorbs that into its knowledge base while maintaining synergy with the environment. Its process phases are given below:

- **Enumeration**: Enumerates all the possible solutions of a task, including new solutions (mutations) that transcend enumerated solutions of the past.
- **Propagation**: During the search of solution for a task, or after the resolution of a task, SIS updates its experience and shares the new information with other peers.
- **Adaptation**: In this phase SIS acquires information about the environment in which it is situated, to adapt the solutions to the environment.
- **Elimination**: Acting according to the information acquired in the previous phases, selects the feasible solutions to solve a task.
- **Concentration**: Finally SIS concentrates on particular solution with its resources.

3. PROPOSED METHOD

Proposed method architecture and its flow diagram are given below.

3.1 ARCHITECTURE

Architecture shows whole process of the work and its two decision cycles. It compares both actual outputs to attain expected best solution.
3.2 FLOW CHART

This flow diagram shows all complete system of the proposed methods.
3.3 PERFORMANCE MEASURE

EXISTING METHOD

Graph 1 comparing actual and expected output of existing method.

PROPOSED METHOD

Graph 2, proposed method comparison difference.

Graph 1 and 2 clearly indicate the effectiveness of the proposed method. In the second graph actual output is almost matches with expected output.

4. CONCLUSION

Proposed methodology not only detect the defects in the software, it also gives suggestion to correct the existing defects because the use of SIS logic. Besides the above mentioned fact continuous monitoring and updating is also performed. In future SIS based testing can be applied from the software specification phase itself.
REFERENCES


