ABSTRACT

Reliability of a software product is the probability of the product working “correctly” under or over a given period of time. It is clear that the reliability of a software system would improve if the number of defects in it is almost zero.

Software testing is the process of executing software in a controlled manner to answer the question: “Does the software behave as specified?” Testing is usually performed for one of the two reasons: defect detection and reliability.

Testing a program consists of providing the program with set of testing inputs (cases) and observing if the program behaved as expected. If the program fails to behave as expected, then the condition under which a failure occurs are noted for debugging and correction.

An adequacy criterion is an essential part of any testing method. The crucial factor in test case design is the set of criteria used to select test case inputs. This set determines what feature of the program under test (PUT) will be covered, hence the extent of the test and the degree of confidence that can be attached to the PUT.

Testers may wish to increase the coverage of the coverable code in the system under test at a faster rate thus allowing a code coverage criterion to be met earlier in the test process.
There is a need to choose the test cases carefully to achieve the necessary coverage while avoiding replication. The question arises next is: how to go about creating good test set with respect to the test adequacy criteria. Since this can be difficult to do by hand, there is a need for Automatic Test Data Generators, also called Dynamic Test Data Generation. Some of the advantages of automation to replace manual software testing and generation of test suites are: reduced testing time, consistent test procedures, reduced quality assurance cost, improved testing productivity and improved product quality.

In this research work, the objective is to conduct “Study of Software Testing and the Evolution of Optimal Method for Quality and Reliability Investigation”. The aim is to contribute to assist the quality of software while optimizing the generation of test cases.

In this regards, the investigation of automated test data generation is studied. The study covers the different methods and strategies applied to test software products. Test adequacy criteria are presented and path coverage test criterion is selected for this study.

The use of Evolutionary Computing (Genetic Algorithms GAs) to automate test data generation and to increase the coverage of the paths in the program code is expounded. Characteristics, strength and limitations of the GA are included.

The role of fitness function and the feedback which it provides to the GA is explained and then a new fitness function (named SMS) is proposed. The SMS based genetic algorithm is thoroughly described.
The main contribution of this work is the increased path coverage within limited time and computation effort. To justify the hypothesis of the research work the tool for automation using SMS-based Genetic algorithms has been built and empirically tested and applied to some samples of small programs for evaluation and validation. Experiments have been carried out to evaluate the efficiency and effectiveness of the proposed SMS based algorithm compared to random testing and other existing algorithms.

Experiments are conducted to study the effect of complexity of a program on the performance of the SMS based algorithm. The performance of the SMS based algorithm under different GA parameter values is also investigated. Furthermore the experiments study the GA performance with different data types such as integers, floats and characters, and also complex data structure such as loops and arrays.

The results confirm that SMS based algorithm can automatically generate test data for path coverage in less number of iterations/generation using less number of individuals/population. The results also prove that all hypotheses defined in the research work are justified.

Efficacy and quality are best served by testing software as early in the life cycle as practical. The later the bug found the higher the cost of fixing it, so it is sound economic to identify and fix the bugs as early as possible.