CHAPTER 3

RESEARCH METHODOLOGY

3.1 RESEARCH MOTIVATION

Software-driven technology has revolutionized all the industrial sectors towards the usage and dependence of software. The development of a software is however, a complex and dynamic task determined by the choice of right technologies and skillset of people. However, the success of a software depends on effectiveness of the test development process. Survey shows that running all the test cases in a test suite may take a great deal of effort. Rothermel et al (2001) has estimated that, for a software comprising of 20,000 lines of code requires around seven weeks to run all its test cases. Hence, there is a great demand for optimization techniques to address the problem of handling the test suite size, as it has evolved into a major resource and cost consumer. To control issues while testing a software, test case selection and evaluation can play a vital role.

Some of the open problems and challenges faced in software testing is listed below. This list is not exhaustive but instead gives an insight to the diverse issues to be addressed (Hyunsook et al 2005, Harman 2007). Some of the major challenges faced during test suite optimization are listed below:

**Stopping Criteria:** In any test suite reduction approach, the stopping criteria is required. Constraints like budget or time may be considered or some criterion may have to be formulated to decide whether optimal testing has
been done. This raises issues like standardization of metrics to address when testing should be stopped.

**Supporting replicability across experiments:** When performing controlled experimentation with software testing techniques several replicability challenges exist such as: test suites utilized by researchers are rarely homogenous, similarly the experiment design and process details may sometimes not be standard. At the outset, high level of replicability in a testing scenario may not guarantee correct aggregation of findings unless there is systematic approach in experimentation.

**Obtaining sample representatives:** The degree of sampling is important as, it has a direct impact on concluding an experiment. Normally, samples considered for experimentation are limited in size and also biased in nature.

**Isolating the effects of individual factors:** Within the testing domain, some factors affect the core of the experimentation process. For example, analysing faults individually or in groups may affect the performance of the testing technique. Similarly, when a program changes in a given version, whether changes are in response to a fault or enhancement or both also affects the assessment of the testing techniques.

This motivates the present work to focus on constructing optimal test suites for data flow testing. Further, effective test measures like test data coverage, fault detection effectiveness and accelerating the test suite reduction process have also been addressed.

### 3.2 RESEARCH OBJECTIVES

The following objectives have been formulated in this research work:
- To identify the current status of the state-of-the-art test suite reduction algorithms.
- To develop algorithms that address test suite reduction.
- To analyse the effectiveness of the test metrics: size, requirement coverage and fault detection in the reduced test suite.
- To evaluate the performance of the proposed algorithms with the state-of-the-art algorithms described in Appendix 2 and Appendix 3.

3.3 RESEARCH CONTRIBUTIONS

The major contributions of this research work include the design and implementation of four algorithms represented in Figure 3.1.

The brief description of the test suite reduction algorithms proposed in this thesis is as follows:

- **An evolutionary algorithm for test suite reduction** - The proposed ABC_TSR (Adapted Bee Colony Test Suite Reduction) algorithm uses the concept of employee and onlooker bees as in the Bee Colony algorithm to probabilistically choose the fittest test cases in the representative test set. Then, the proposed algorithm is empirically evaluated with the classical HGS algorithm and BOG algorithm. However this algorithm focuses only on the test suite size and requirement coverage metrics.

- **A greedy algorithm for test suite reduction** - The proposed CBTSR (Coverage Based Test Suite Reduction) algorithm
selects a subset of test cases that exercises the given set of requirements with respect to data flow adequacy criterion. The proposed algorithm uses simple matrix computation and the effectiveness of the proposed algorithm is evaluated using the existing HGS and BOG algorithms. The proposed algorithm also focuses on two metrics: test suite size and requirement coverage.

![Diagram of test suite reduction algorithms]

**Figure 3.1 Test suite reduction algorithms developed**

- A graph based algorithm for test suite reduction - Kruskal’s algorithm has been adapted to construct the representative set with respect to data flow coverage criterion, while retaining a high percentage of the original test suite’s fault detection effectiveness. The proposed AKTSR (Adapted Kruskal’s Test Suite Reduction) algorithm is evaluated with the existing HGS and BOG algorithms for the test metrics: size, requirement coverage and fault detection density.
- **Test suite reduction using a data mining algorithm** – The MFTS (Maximal Frequent Test Set) algorithm selects maximum frequent test cases in a test suite. This algorithm uses dynamic minimum support to select maximum frequent test cases that ensures requirement coverage of the subject program. The experimental studies compare the relative performance and effectiveness of the proposed reduction algorithm with the classical HGS and BOG algorithms for the test metrics: test suite size, requirement coverage, fault detection density and execution time.

### 3.4 SCOPE OF THE THESIS WORK

The main focus of this research work is to propose and develop algorithms for test suite minimization. This thesis work has been done in the perspective of

- Structural testing technique called data flow testing has been used. It considers the possible interactions between definition and use of variables referred to as Define/Use (DU) pairs.

- The recording of DU pairs in a program have been done using Rapps & Weyuker (1985) dataflow criterion. These DU pairs refer to the requirements that are to be covered by each of the subject programs. The DU pairs used as requirements for the ten programs have been hand-instrumented.

- For all the ten subject programs hand seeded faults of equal severity has been injected.
All the four proposed algorithms have been implemented in Java and tested against ten toy programs.

The scope of this work is limited to subject programs consisting of 1 to 4 variable interactions.

3.5 SUMMARY

This chapter has presented the motivation behind this research work along with the research objectives. The research contributions have also been debriefed. Finally, the scope of the present work pertaining to test suite reduction has been summarized.

The next chapter describes two algorithms for test suite reduction. The first algorithm proposed is based on an evolutionary algorithm that focuses on test suite reduction based on the fitness value. On the contrary, the second algorithm is a greedy approach using simple matrix computation for test suite reduction. Both the proposed algorithm have been evaluated for the test metrics: test suite size and requirement coverage.