Chapter 1

Introduction

In this chapter we give a brief introduction and motivation for the research work presented in this thesis. After brief motivation, we commence by introducing the problems in random testing. We then describe the alternative approaches to overcome these problems, followed by our research goals and contributions. At the end of the chapter, we give the structure of the thesis.

1.1 Motivation

Software is a very important component of the system essentially required to accomplish the given task. Some softwares are developed for use in simple day to day operations whiles others are for highly complex processes in specialized fields like education, health and medicine, science and technology, business, commerce and industry, information and communication, defense, environment and safety etc. The margin of error in the mission-critical and safety critical systems is so small that a minor fault can incur huge cost to the organization [45]. Therefore, software companies leave no stone unturned to ensure the reliability and accuracy of the software. This dissertation is a humble contribution aiming to reduce the overall cost of software testing by devising new improved and highly effective software testing techniques.

1.2 The Problem

Exhaustive testing of software is not always possible and the problem of selecting a test data set, from a large or often infinite domain is often confronted. Test data set, as a subset of the whole domain, is carefully selected to test the given software. An adequate test data set is a crucial factor in any testing technique because it represents the whole domain for evaluating the structural and/or functional properties [57], [44]. Manual test data set generation is a time-consuming and laborious exercise [49]; therefore, automated test data set generation is always preferred. Test data generators are classified in to Path-wise, Goal-Oriented, Intelligent and Random [89]. Random test data generation involves generating test data set randomly from the whole domain. Unlike other approaches Random technique is simple, widely applicable, easy to implement, faster in computation, with minimum overhead and free from bias [25]. It is stated by -------- that, “Random testing is simple and well-known technique which can be remarkably effective at software bugs [DART].

Despite the benefits of random testing, its simplistic and non-systematic nature exposes it to high criticism [88]. Myers & Sandler [59] mentioned it as, “Probably the poorest methodology of all random-input testing...”. According to -------- this statement is based on intuition and lacks any experimental evidence [---]. However it motivated the interest of research community to evaluate and improve random testing. Adaptive random testing [13], Restricted Random Testing [10], Feed-back directed Random Testing [75], Mirror Adaptive Random Testing [14] and Quasi Random Testing [18] are a few of the enhanced random testing techniques with higher fault finding ability.

Random testing is also considered weak in providing high code coverage [63], [30]. For example, in random testing when the conditional statement *“if (x == 25) then …”* is exposed to execution then there is only one chance, of the “*then...*” part of the statement, to be executed out of 232. If x is an integer variable of 32 bit value [40].

Random testing is no exception when it comes to the complexity of understanding and evaluating test results. Modern testing techniques simplifies results by truncating the lengthy log files and displaying only the fault revealing test cases in the form of unit tests. However, efforts are required to show the test results in more compact and user-friendly way.

1.3 Research Goals

The goals of the research study are to develop new techniques for automated testing, based on random strategy, with the aim to achieve the following objectives:

1. To develop an automated random testing technique with the capability to generate more fault-revealing test data.
2. To develop a framework for finding the faults, fault domains and presentation of results on a graphical chart within the specified lower and upper bound.
3. To develop a specialized automated random testing technique with focus on increase in code coverage along with generation of more fault-revealing test data.

1.4 Contributions

The main contributions of the thesis research are stated below:

1.4.1 Dirt Spot Sweeping Random Strategy

The efficiency decreases when the failures lie in contiguous locations across the input domain. To overcome the problem, a new automated technique: Dirt Spot Sweeping Random (DSSR) strategy was developed. It is based on the assumption that unique failures reside in contiguous blocks and stripes. When a failure is identified, the DSSR strategy selects neighboring values for the subsequent tests. Resultantly, selected values sweep around the failure leading to the discovery of new failures in the vicinity. Results indicated that the DSSR strategy will have a profound positive impact on the failure-finding ability of R and R+ strategies.

1.4.2 Automated Discovery of Failure Domain

The automated random strategies of software testing based on the presence of point, block and strip fault domains lack the capability of full automation for the discovery and evaluation of fault domains. A fully automated testing strategy named Automated Discovery of Failure Domain (ADFD), was developed which finds the faults and the fault domains in a given SUT and provides visualization of the identified pass and fail domain. The research study includes description of the ADFD strategy, its implementation in YETI and its illustration with the help of an example. The experimental results showed that ADFD strategy automatically performs identification of faults and fault domains along with their representation on graphical chart.

1.4.3 Invariant Guided Random+ Strategy

A more efficient random test strategy named Invariant guided Random+ Strategy (IGRS) was developed in this research study. IGRS is an extended form of Random+ strategy guided by software invariants. Invariants from the given SUT are collected by Daikon— an automated invariant detector, which reports likely invariant, prior to testing and add these to the SUT as assertions. The IGRS strategy, implemented in YETI, generates values in compliance with the added assertions. Experimental results indicated improved features of IGRS in the form of higher code coverage and identification of subtle errors that pure Random and Random+ strategies were either unable to accomplish or took a large duration.

1.5 Structure of the Thesis

The rest of the thesis is organized as follows: In Chapter 2, a thorough review of the relevant literature is given. We commence by discussing a brief introduction of software testing and highlighted various techniques and types of software testing followed by automated random testing and the testing tools. Chapter 3 includes Dirt Spot Sweeping Random (DSSR) strategy based on sweeping faults from the clusters in the input domain. Chapter 4 describes the second automated random strategy focusing on dynamically finding the faults and their domains along with graphical representation. Chapter 5 presents the third strategy focusing on quick identification of faults and increase in coverage with the help of assertions; Finally Chapter 7 summarizes contributions of the thesis, discusses the strength and weaknesses of the study, and suggests avenues for future work.