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 everywhere. In our world today, software flies spacecraft, monitors power plants, manages stock exchange, assist surgeries, drives cars and design graphics. The margin for error in these mission-critical and safety-critical systems is so small that a minor fault can incur huge cost to the economy and miseries to the mankind [45]. Therefore, software development companies leave no stone unturned to ensure the reliability and accuracy of the software. This dissertation is a step further towards the reduction in overall cost of software testing by devising new improved and highly effective software testing techniques.

1.2 The Problems

In software testing, one is often confronted with the problem of selecting a test data set, from a large or often infinite domain, as exhaustive testing is not always applicable. Test data set is a subset of domain carefully selected to test the given software. Finding an adequate test data set is a crucial process in any testing technique as it aims to represent the whole domain and evaluate the given system under test (SUT) for structural 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 generates test data set randomly from the whole domain. Unlike other approaches Random approach is simple, widely applicable, easy to implement in an automatic testing tool, fastest in computation, no overhead in choosing inputs and free from bias [25].

Despite the benefits random testing offers, its simplistic and non-systematic nature expose it to high criticism [88]. Myers & Sandler [59] mentioned it as “Probably the poorest methodology of all is random-input testing...”. Where this statement is based on intuition and lacks any experimental evidence, 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 few of the enhanced random testing techniques aiming to increase its 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 display 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 overall goal of this thesis is to develop new techniques for automated testing based on random strategy that addresses the above-mentioned problems. Particularly,

1. We aim to develop an automated random testing technique that is able to generate more fault-revealing test data. To achieve this we exploit the presence of fault clusters found in the form of block and strip fault domains inside the input domain of a given SUT. Thus we are able to find equal number of faults in fewer numbers of test cases than other random strategies.
2. We aim to develop a novel framework for finding the faults, their domains and the presentation of obtained results on a graphical chart inside the specified lower and upper bound. It considers the correlations of the fault and fault domain. It also gives a simplified and user-friendly report to easily identify the faulty regions across the whole domain.
3. We aim to develop another automated testing technique which focuses on increase in code coverage and generation of more fault-revealing data. To achieve this we utilizes Daikon —an automated invariant detector that reports likely program invariant. An invariant is a property that holds at certain point or points in a program. With these invariants in hand we can restrict the random strategy to generate values around these critical points. Thus we are able to increase the code coverage and quick identification of faults.

1.4 Contributions

To achieve the research goals described in Section 1.3, we make the following specific contributions:

1.4.1 Dirt Spot Sweeping Random Strategy

Random testing is a simple and effective technique to find failures in complex programs. How- ever, its efficiency reduces when the failures lie in contiguous locations across the input domain. To overcome the deficiency, we developed a new automated technique: Dirt Spot Sweeping Random (DSSR) strategy. 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. To evaluate the effectiveness of DSSR strategy a total of 60 classes (35,785 lines of code), each class with 30 x 105 calls, were tested by Random (R), Random+ (R+) and DSSR strategies. T-Test analysis showed significantly better performance of DSSR compared to R strategy in 17 classes and R+ strategy in 9 classes. In the remaining classes all the three strategies performed equally well. Numerically, the DSSR strategy found 43 and 12 more unique failures than R and R+ strategies respectively. This study comprehends that DSSR strategy will have a profound positive impact on the failure-finding ability of R and R+ testing.

1.4.2 Automated Discovery of Failure Domain

There are several automated random strategies of software testing based on the presence of point, block and strip fault domains inside the whole input domain. As yet no particular, fully automated test strategy has been developed for the discovery and evaluation of the fault do- mains. We therefore have developed Automated Discovery of Failure Domain, a new random test strategy that finds the faults and the fault domains in a given system under test. It further provides visualization of the identified pass and fail domain. In this paper we describe ADFD strategy, its implementation in YETI and illustrate its working with the help of an example. We report on experiments in which we tested error seeded one and two-dimensional numerical programs. Our experimental results show that for each SUT, ADFD strategy successfully performs identification of faults, fault domains and their representation on graphical chart.

1.4.3 Invariant Guided Random+ Strategy

Acknowledgement of random testing being simple in implementation, quick in test case generation and free from any bias, motivated research community to do more for increase in performance, particularly, in code coverage and fault-finding ability. One such effort is Random+ that is an ordinary random testing technique with addition of interesting values (border values) of high preference. We took a step further and developed Invariant Guided Random+ Strategy (IGRS). IGRS is an extended form of Random+ strategy guided by software invariants. Invariants from the given software under test are collected by Daikon— an automated invariant detector that reports likely invariant, prior to testing and added to the list of interesting values with high preference. The strategy generates more values around these critical program values. Experimental result shows that IGRS not only increase the code coverage but also find some subtle errors that pure Random and Random+ were either unable or may take a long time to find.

1.5 Structure of the Thesis

The rest of the thesis is organized as follows: In Chapter 2, we give a thorough review of the relevant literature. We commence by discussing a brief introduction of software testing and shed light on various techniques and types of software testing. Then, we extend our attention to automated random testing and the testing tools using random technique to test softwares. In Chapter 3, we present our first automated random strategy Dirt Spot Sweeping Random (DSSR) strategy based on sweeping faults from the clusters in the input domain. Chapter 4 describes our second automated random strategy which focus on dynamically finding the fault with their domains and its graphical representation. Chapter 5 presents the third strategy that focus on quick identification of faults and increase in coverage with the help of literals; Finally, in Chapter 7, we summarize the contributions of this thesis, discuss the weaknesses in the work, and suggest avenues for future work.