Application of Search-based Software Testing to Non-functional system properties.

A Validated Framework

Nitin Parasa

Faculty of Computing
Blekinge Institute of Technology
SE-371 79 Karlskrona, Sweden
This thesis is submitted to the Faculty of Computing at Blekinge Institute of Technology in partial fulfillment of the requirements for the degree of Master of Science in Software Engineering. The thesis is equivalent to 20 weeks of full time studies.

Contact Information:
Author(s):
Nitin Parasa
E-mail: nipa15@student.bth.se

University advisor:
Bogdan Marculescu
Department of MSE

Faculty of Computing
Blekinge Institute of Technology
SE–371 79 Karlskrona, Sweden

Internet : www.bth.se
Phone : +46 455 38 50 00
Fax : +46 455 38 50 57
Abstract

**Context** The importance of testing non-functional properties of the system is growing steadily. Complexity factor of the software is growing proportionally with the growing demands and hence attributes like performance, energy consumption and reliability are proving to be very crucial. Optimizing the software with respect to these properties simultaneously with the functional properties has been found to be a challenge. Search-based Software testing automates this process by using different meta-heuristic techniques. It assures the generation of large number of test cases at a minimal cost. Carrying out testing in this context requires lot of expertise and the aid of a highly flexible approach. There is a strong need of a guide that helps the practitioners(testers) and researchers optimize the non-functional properties using Search-based software testing.

**Objectives** The objective of the work presented in this thesis is to, first, investigate the non-functional properties, challenges encountered and approaches/suggestions by the practitioners on the application of Search-based software testing in academia and industry. Second objective is to map all the information into a conceptual/theoretical framework that could be used by Search-based software testing practitioners for optimizing the non-functional system properties.

**Methods** A qualitative approach has been employed for this thesis work. A literature review with snowball sampling as the search approach was conducted to collect the information regarding the different kinds of systems being tested, the non-functional system properties being optimized, challenges encountered and the tools used for this purpose. Semi-structured interviews are conducted as a part of the validation process and generalizing the results obtained. A total of 9 interviews were conducted. Thematic analysis technique has been used to analyze the collected data.

**Results** As a result of conducting this research, different dimensions forming the framework have been investigated. The overall result of this study is the formulation of a framework and that has been validated by conducting interviews. The framework consists of 16 challenges related to the field of Non-functional Search-based software testing.
Conclusions It is found out that Search-based testing for non-functional properties has not been extensively applied in the industry. It has been suggested, used and applied in academia for the most part. Several factors influence the selection of non-functional properties for optimization. Most of the challenges being faced in this subject are inclined towards three areas in Search-based testing. Performance, execution time and energy consumption are three most popularly tested attributes. Further research could be done wherein the framework generated could be put to use by different practitioners and researchers to find out interesting things.

Keywords: Non-functional system properties, Search-based Software Testing, Search-based Software Engineering, Meta-heuristic techniques, Framework.
There are very few events in life that are cherished forever. This master thesis is one among them. Regardless of the problems faced the end goal has been sought out well. First of all, I would like to thank my supervisor Bogdan Marculescu for his unconditional support, timely advice and frequent reviews. I have had a wonderful chance of learning a lot from his rich expertise in research and writing.

I would like to thank the experts Wasif Afzal, Simon Poulding, Shin Yoo, Ronak Shah and others who took time and helped in validating this research. Their passionate participation and input had a great impact on this research.

I would like to express my deepest gratitude to my parents, Mr. P.N.V Suresh Kumar and Mrs. P. Nischala, brother P. Nikhil, my aunt Mrs. A. Neeraja Roy and friends for their everlasting support, encouragement and for having had constant faith in me.

Thank you all.

Nitin Parasa

"Not many people find writing easy. It is important to emphasize that all writing has difficulties and it is not something that only other people can do. Writing is something that most of us can do if we persevere."

Chris Hart
## Contents

<table>
<thead>
<tr>
<th>Section</th>
<th>Pages</th>
</tr>
</thead>
<tbody>
<tr>
<td>Abstract</td>
<td>i</td>
</tr>
<tr>
<td><strong>1 Introduction</strong></td>
<td>1</td>
</tr>
<tr>
<td>1.1 Software Testing - An Overview</td>
<td>1</td>
</tr>
<tr>
<td>1.2 Problem Description</td>
<td>2</td>
</tr>
<tr>
<td>1.3 Thesis Structure</td>
<td>3</td>
</tr>
<tr>
<td><strong>2 Background and Related Work</strong></td>
<td>4</td>
</tr>
<tr>
<td>2.1 Search-based Software Testing</td>
<td>4</td>
</tr>
<tr>
<td>2.1.1 Overview</td>
<td>4</td>
</tr>
<tr>
<td>2.1.2 Commonly used algorithms</td>
<td>6</td>
</tr>
<tr>
<td>2.1.3 Where is Search-based software testing applied?</td>
<td>9</td>
</tr>
<tr>
<td>2.2 Non-Functional System Properties</td>
<td>9</td>
</tr>
<tr>
<td>2.2.1 Overview of Requirements Engineering</td>
<td>9</td>
</tr>
<tr>
<td>2.2.2 Non-Functional Requirements</td>
<td>10</td>
</tr>
<tr>
<td>2.3 Related Work</td>
<td>12</td>
</tr>
<tr>
<td><strong>3 Method</strong></td>
<td>14</td>
</tr>
<tr>
<td>3.1 Research Purpose</td>
<td>14</td>
</tr>
<tr>
<td>3.1.1 Objectives</td>
<td>14</td>
</tr>
<tr>
<td>3.2 Research Questions and Instrument</td>
<td>15</td>
</tr>
<tr>
<td>3.3 Research Method</td>
<td>16</td>
</tr>
<tr>
<td>3.4 Literature Review</td>
<td>16</td>
</tr>
<tr>
<td>3.4.1 Snowballing as a Search Approach</td>
<td>17</td>
</tr>
<tr>
<td>3.4.2 Snowballing Procedure</td>
<td>18</td>
</tr>
<tr>
<td>3.5 Validating the Framework</td>
<td>23</td>
</tr>
<tr>
<td>3.6 Semi-Structured Interviews</td>
<td>24</td>
</tr>
<tr>
<td>3.6.1 Interview Subjects</td>
<td>24</td>
</tr>
<tr>
<td>3.6.2 Interview Design</td>
<td>26</td>
</tr>
<tr>
<td>3.6.3 Interview Instrument</td>
<td>26</td>
</tr>
<tr>
<td>3.7 Data Analysis</td>
<td>27</td>
</tr>
<tr>
<td><strong>4 Results of Literature Review</strong></td>
<td>31</td>
</tr>
<tr>
<td>4.1 Software systems under test</td>
<td>31</td>
</tr>
</tbody>
</table>
List of Figures

2.1 A search for local optimum (Source: McMinn [50]) . . . . . . . . . 7
2.2 Finding global optimum (Source: McMinn [50]) . . . . . . . . . . 7
2.3 Search space for Genetic Algorithms (Source: McMinn [50]) . . . 8
2.4 Example for crossover (Source: Harman et al. [25]) . . . . . . . . 8
2.5 Types of Non-Functional Requirements . . . . . . . . . . . . . . . 12

3.1 Research Time line . . . . . . . . . . . . . . . . . . . . . . . . . 16
3.2 Snowballing Process Flow . . . . . . . . . . . . . . . . . . . . . . 19
3.3 Start-set Selection Process . . . . . . . . . . . . . . . . . . . . . . 21
3.4 Steps followed for Thematic Analysis . . . . . . . . . . . . . . . . 28
3.5 Themes formed using Nvivo . . . . . . . . . . . . . . . . . . . . . 30

5.1 Researchers in Search-based testing domain . . . . . . . . . . . 40
5.2 Practitioners in Non-functional testing domain . . . . . . . . . . 41
5.3 Thematic map of Systems under test . . . . . . . . . . . . . . . . 42
5.4 Thematic map of Non-functional requirements tested using SBST 43
5.5 Challenges encountered employing NFSBST . . . . . . . . . . . . 48

6.1 Overview of the Framework . . . . . . . . . . . . . . . . . . . . . 59
6.2 Non-functional Search-based Software Testing Framework (Part1) 61
6.3 Non-functional Search-based Software Testing Framework (Part2) 62
List of Tables

2.1 Non-Functional Properties and their Interpretation ........................................ 10
3.1 Keywords ........................................................................................................ 20
3.2 Interviewee Details (Demographics) ................................................................. 25
4.1 Software Systems Under Test ........................................................................... 32
4.2 Meta-heuristic Techniques ................................................................................ 33
4.3 Non-functional properties reported in the literature ......................................... 34
4.4 Tools Used ......................................................................................................... 38
5.1 Participants response on importance of the Non-functional properties ............ 47
6.1 Software Systems Under Test ........................................................................... 56
6.2 Challenges and Approaches while employing NFSBST .................................... 57
6.3 Articles used in the framework .......................................................................... 60
Abbreviations

NFP  Non-functional Property
NFR  Non-functional Requirement
SBST Search-based Software Testing
SBSE Search-based Software Engineering
NFSBST Non-functional Search-based Software Testing
Chapter 1

Introduction

Over the past three decades or so, the field of Software Testing has witnessed a tremendous rise in the level of importance being given to address the non-functional system properties. Testing is regarded as one of the most resource consuming tasks in a software project [52]. The main goal of testing related activities is to make sure that the requirements are satisfied. As the non-functional requirements are emerging, it is proving to be a difficult task for the companies to keep up the high standards.

1.1 Software Testing - An Overview

“Software is pervasive, which rises the value of testing it” [57]. Software Testing is the process or a series of activities contributing to delivering a software that is doing what it is designed to do and the converse [57]. The quality of a product can be determined by the extent to which the software product developed, meets the requirements specified [29]. Verification and Validation of the product must be performed throughout the development process.

Verification is the process of inspecting if the development of a system is satisfying the given conditions that have been mentioned at the start of the process [29]. Validation is the process of inspecting if the system has satisfied all the specified requirements at the end of the development process [29].

Many authors have highlighted different types of testing approaches [35] [48] [15]. They include Black-box testing, White-box testing, Stress testing, Security testing, Database testing, System testing, Integration testing, Unit testing, Acceptance testing, etc. But there are three primary types of testing techniques [57] [35]. They are:

1. **White-box Testing**: It is also referred to as Structural testing or Glass-box testing which relies on the internal knowledge of the system to carry out the testing process [15]. The main goal of this form of testing is to target a few areas of the system like the program branches and specific statements [48]. It highlights the importance of the internal structure of the software system.
2. **Black-box Testing**: It is also referred to as Functional testing or behavioral testing which focuses on the core functionality of the system. System requirements play a major role in this type of testing process [15]. It does not stress the importance of having knowledge on the internal knowledge of the system under test. It highlights the importance of the external behavior of the system [48].

3. **Grey-box Testing**: It is a combination of both Black-box and White-box testing techniques. It requires the tester to possess limited knowledge both on the internal structure and the fundamental behaviors of the system.

That’s just not it. In the world of software testing, apart from these terms, we often come across the terms Manual testing and Automated testing. A software can be generally tested using both these testing strategies. It is said that both these strategies are complementary to each other. They have their own drawbacks which the other strategy addresses [42]. Manual testing helps in capturing very special test cases which cannot be captured using automated testing. Automated testing helps in capturing extensive coverage of test cases which is very difficult through manual testing [42].

Torkar and Mankefors [65] found that 60 percent of the developers would neglect the verification and validation of the product in case of resources shortages in the development process. “We need the new software application sooner than that.” [11]. Today’s managers are being forced to develop the products with shrinking deadlines and minimal resources. To address this tricky issue, the concept of automation came into existence. Automation is the process of carrying out things with minimal human interaction. In this context, Automated software testing is defined as [11] the automation of software activities like the test case execution, test result and requirements verification with the use of automated tools.

Likewise, different kinds of testing techniques have been developed and being used by both the industry and the researchers in academia. Some of them include Evolutionary testing, Search-based software testing, etc. Search-based software engineering is one such approach that has proved to be successful for over ten years [25]. It adopts the concepts of Search-based software engineering. The dynamic nature of SBSE, i.e. using the fitness function to carry out the optimization processes makes it a very effective tool to deal with the non-functional system properties [72]. A brief description on Search-based software testing is provided in chapter 2.

### 1.2 Problem Description

Afzal et al. [2] mention the importance of carrying out another study that concentrates on other non-functional system properties as their main concentra-
Chapter 1. Introduction

The motivation was only on Usability, Safety, Execution time, Security and Quality of Service (Qos). This motivated the author to further extend the existing work. Existing work focused on finding out the meta-heuristics (refer chapter 2) and fitness functions (refer chapter 2) used to test these properties. Challenges encountered while employing these techniques to test the properties has also been one of the core components. What was actually missing in the existing work was a set of approaches/ suggestions available from different sources which could potentially solve the challenges identified. It is also important to identify different software systems to which Search-based software testing has been applied and correlate them with a set of non-functional properties and list out a set of challenges and approaches / suggestions to solve them. Hence, for this research finding a way of presenting this data has been of prime concern. That idea eventually lead to the formulation of a framework which provides information about Non-functional Search-based Software Testing (NFSBST) in general. This could eventually help the target audience of the framework in identifying various kinds of information in this area. To achieve this goal, several questions need to be answered. Further down in the document, relevant details are provided which eventually lead to the formulation of the framework.

1.3 Thesis Structure

This thesis is structured as follows; chapter 1 (Introduction) and chapter 2 (Background and Related Work) focus on introducing the topic to the reader and also provides the relevant background and related work done that has been done. Next, the Research Methodology has been clearly described in Chapter 3 of this document. Chapter 4 (Results of Literature Review) - provides a detailed description about the results that have been obtained from the existing literature, chapter 5 (Results and Analysis of Semi-structured interviews) - gives a detailed explanation of the results that have been obtained from the interviews and how interesting information has been found using the analysis techniques, form the Results and Analysis parts of the document. chapter 6 (Framework) - gives a brief overview of the framework along with the diagrammatic representations, chapter 7 (Discussions) discuss the overall findings of the research conducted and the threats to validity. Finally, in chapter 8 (Conclusions and Future Work) the summary of the results and the contributions of this research.
Chapter 2

Background and Related Work

This research revolves around the concepts of Search-based software testing applied for the non-functional systems properties, different software systems under test, the kind of challenges the practitioners face while applying these techniques and different tools that are currently in use for carrying out these tasks. Hence, this chapter introduces the background required for the study along with the previous works published in relation to this research.

2.1 Search-based Software Testing

2.1.1 Overview

Search-based software testing has evolved from Search-based Software Engineering. SBSE is the approach to software engineering by the application of search based optimization. It reformulates the software engineering problems as ‘search problems’ [23]. A search problem is that where an optimal or sub-optimal solution is found from the candidate solution set. This is guided by a mathematical equation called the fitness function that determines if a solution is better or worse [25].

There are two key ingredients to get started with SBSE [25]. They are:

1. Representation of the problem.
2. Fitness Function

There exists a suitable representation of a problem in this context. Few metrics are associated with the problems and these form the candidate solution set for the fitness function. A fitness function is a mathematical equation of the representation to the search problem [25]. Search-based software testing uses the concepts of SBSE in efficiently automating the process of testing.

Search-based software testing is the application of the meta-heuristic techniques for generating the test data automatically [49]. It is a tool which can be applied both manually and automatically. Most of the research focuses on the
automated Search-based software testing. As discussed above the manual process of testing the software is very laborious and costly. These programs tend to be unreliable and seems to be difficult to exercise the deeper features of the software. Researchers found out that usage of meta-heuristic techniques might address these problems [49]. The meta-heuristic techniques work by encoding the solution so that the search space could be manipulated. This encoding is done by spreading to the neighbors with similar solution space. An objective function is responsible for all these actions, the search spreading to different neighbors and accessing the different parts of the solution space to find different test cases.

Different meta-heuristic techniques are in use [49] [50] [18]. Most of them have been inspired from biology and different techniques keep on evolving. A few of them are listed below.

1. **Evolutionary Algorithms**: Evolutionary algorithms are those optimization techniques that work on the principle of Darwinian Evolution. The solutions are identified as chromosomes. The evolution of chromosomes happens due to crossover actions, mutations and natural selection. Nowadays the application of pure Evolutionary algorithms is not very effective search method [39].

2. **Tabu Search (TS)**: This technique has been very popular in solving the combinatorial optimization problems ranging from scheduling to space planning. In it simplest form, if \( S(x) \) is a set of moves that the function makes in the space and \( T \) is a subset of \( S \) called as the tabumoves. Initially the tabumoves are set to zero and as the iterations starts the solutions are recorded. If a new optimum solution is found it is transferred to the tabulist and then repeats the iterations. From there the search algorithm works its way to find the best solution [17].

3. **Memetic Algorithms**: Memetic algorithms are an extension of evolutionary algorithms but they apply different search processes to refine the solutions identified. They might for example, employ hill climbing to improve the fitness function [39]. They are a combination fo both evolutionary algorithms and local search algorithms.

4. **Variable Neighborhood Search**: This kind of meta-heuristics solve most frequently the combinatorial optimization problems with the basic idea of moving to the next neighborhood in a systematic manner in the local search [55].

5. **Guided Local Search**: This technique is also used to solve combinatorial problems. The search for solution takes place in several iterations [67].

6. **GRASP**: It stands for Greedy Randomized Adaptive Search Procedure. It repeatedly improves the solutions by local search. For every iteration both
local search and GRASP are implemented to find the local optima. All the local optima are collected and returned as the final solution of GRASP [44].

7. **Ant Colony Optimization (ACO):** This optimization technique has been inspired from the behavior of some ant species. Ants leave impressions of the path that they want other members of the colony to follow. In the same way, in ACO, artificial ants build the solutions to the problems. The quality of the solutions is communicated similar to what the real ants follow. ACO is an iterative algorithm where the solutions are built traversing from vertex to vertex. Any vertex in the graph is not visited twice. At the end of each iteration, depending on the quality of the solutions constructed by the artificial ants, pheromone (paths) values are modified so that ants in the future iterations build solution closer to the good ones [10].

8. **Particle Swarm Optimization (PSO):** In this technique a set of solutions are used in the search space. Each of the solution tends to optimize according to the algorithm being used. Later on we can observe that the solutions cluster either to form a maxima or minima. It requires simple mathematical operators to form the fitness function and it is computationally inexpensive [34].

### 2.1.2 Commonly used algorithms

Some of the commonly used algorithms are briefly described for a better understandability of this research. Any Search-based application needs to use a fitness function to guide the search. Random search is one search algorithm that does not use the fitness function [25]. To find out higher quality solutions, the need of a function is necessary to guide the search. One of the most commonly used algorithms is Hill Climbing [25].

**Hill Climbing** - Hill Climbing selects a point in the search space and then examines for the possible solutions that exist in the neighborhood. Neighborhood [50] in this context represents the solutions in the search space that are similar but different in some aspects. The search moves throughout the space to find out solutions with improved fitness. New neighborhoods are constantly inspected until a good solution is found. Such a solution is termed as *locally optimal* [25] but it is not the global optimum (refer figure 2.1). To find out the global optimum (refer figure 2.2) the search would be restarted and factors like the resource available to carry out this are to be considered.
Simulated Annealing - This has been first proposed by Kirkpatrick et al. [36], where attempts are made to improve one solution. In Simulated Annealing the search is not always restarted each and every time like what happens in Hill Climbing. When it comes to finding the global optimum, the technique it follows is to accept even the poor solutions outside the neighborhood and continue until it reaches the least optimal solution. Then, it starts to behave identically to Hill Climbing [50].

Hill Climbing and Simulated Annealing are referred to as local searches [25]. On the other hand, Genetic Algorithms and its versions are referred to as global searches in the sense that they sample many points in the search space offering more reliable and optimal values.
**Genetic Algorithms** - The solutions here are referred with different names. A *current population* is the set of candidate solutions currently under consideration and successive solutions are referred as *generation*. Inspired by Darwin Evolution the candidate solutions are referred as *chromosomes* [25]. Typical candidate solutions are represented using binary 1’s and 0’s. The process of searching for the solutions initially starts with a randomly generated population (refer figure 2.3) and evaluating each of them for fitness. After that, a *crossover* is performed to recombine the chromosomes to form offspring chromosomes. Refer to figure 2.4 for better understanding. The reason for performing a crossover is to diversify the search into new neighborhoods of the search space. This process is repeated where each generation of solutions are evaluated for fitness and the resources allocated to the search are exhausted [50].

![Search space for Genetic Algorithms](Source: McMinn [50])

**Figure 2.3:** Search space for Genetic Algorithms (Source: McMinn [50])

![Example for crossover](Source: Harman et al. [25])

**Figure 2.4:** Example for crossover (Source: Harman et al. [25])

A lot has been achieved in Search-base Engineering that the usage of fitness function can be scaled to any level. It is always recommended that a single fitness function and simple representation of the problem should be used to run the first experiments [25]. Once the results prove to be promising then more advanced
techniques could be used that helps in finding even more reliable solutions. Some of the techniques are Multi-objectives and Co-evolution [25].

2.1.3 Where is Search-based software testing applied?

Search-based Testing has been applied in almost all fields of testing a software product. These fields include the following [50]:

- **Functional Testing**: Search-based testing in this field could be well understood looking at the experiment conducted by Buehler and Wegener [7] at the Daimler AG company on the automated parking system in the car.

- **Structural Testing**: This is the area that has attracted many, towards the application of Search-based software testing. The software under test is inspected and then the inputs are given to the software from the search space suggestions. Finally, the comparison is done with respect to the path or branch coverage that is most optimum for the software under test [50].

- **Grey-box Testing**: The application of Search-based testing to this field includes the Assertion Testing and Exception condition testing [49].

- **Temporal or Non-Functional Testing**: The application of Search-based testing in this area has been comparatively less. Most of the research was based on the execution times of the software under test. Research that has been conducted on the application of Search-based testing to this field is available in [2] [50] [49].

2.2 Non-Functional System Properties

2.2.1 Overview of Requirements Engineering

Requirements engineering is one of the main phases of the software development life cycle [19]. This phase starts the development lifecycle by gathering the requirements from the customers. Customers can be generally of different types, hence, the kinds of projects also change depending on the customer being targeted. Generally, there are three kinds of projects [58] [66]. They are: 1) Bespoke or Small-scale projects 2) Large-scale or Market driven projects and 3) Very Large-scale projects. Irrespective of the scale of the project, assuring the quality of the end product is of prime concern. Hence testing the product with respect to these requirements is very crucial. In a typical project, two kinds of requirements exist. They are:

- **Functional Requirements**: It specifies what the system should do. This kind of requirements deals with the symbolic contents of the software. They specify the behavior or functionality of the software product.
• **Non-Functional Requirements**: In Software engineering, those requirements which describe how the system should behave or perform functions in different environments are termed as non-functional requirements. These kind of requirements try to complement the functional properties of the system. They are difficult to test and they are often inspected subjectively [64].

Other types of requirements like the Business Requirements, Market Requirements, User-Interface Requirements form the subset of the two main types of software requirements. For this study, the prime concern is on Non-functional system properties. These properties are difficult to interpret and analyze [64]. Many factors influence these phenomenon (interpretation, testing) which are discussed later on in this document. These properties are very subjective and when it comes to reality, they behave very strangely making it very difficult to work on them.

### 2.2.2 Non-Functional Requirements

Non-Functional Requirements (NFR) also known as quality requirements, are vital for the success of the software. Finding the right balance between both the functional and the quality requirements is very important [4]. If these requirements are not satisfied, poor product quality, inconsistencies in the software are inevitable. NFR’s can be subjective and relative making different people interpret and evaluate them in different ways. They can also be interactive in nature wherein trying to achieve one NFR can affect the goals of another NFR both positively or negatively [8]. It is very important to deal with them effectively. These system properties are of several kinds and many researchers have made attempts in classifying them [31] [30] [14] [60] and finding different categories of such properties. Collective information of these properties is presented with a detailed description of what each of these properties mean. A mind map of these properties in figure 2.5 is also available. Different Non-Functional properties and their interpretation in relation to software development is presented in Table 2.1. The terms Non-functional systems properties, Non-Functional requirements, quality requirements are used interchangeably throughout the course of this document.

<table>
<thead>
<tr>
<th>Non-Functional Property</th>
<th>Interpretation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Usability</td>
<td>It is the ease with which the targeted users can use the system.</td>
</tr>
<tr>
<td>Safety</td>
<td>It is the degree to which the accidental harm is detected and prevented.</td>
</tr>
<tr>
<td>Robustness</td>
<td>It is the ability of the system responding effectively to different abnormal system conditions.</td>
</tr>
</tbody>
</table>
Chapter 2. Background and Related Work

<table>
<thead>
<tr>
<th>Quality Dimension</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Integrity</td>
<td>The degree to which the intentional and unauthorized access to the system is restricted.</td>
</tr>
<tr>
<td>Efficiency</td>
<td>It refers to the effective usage of all the available resources.</td>
</tr>
<tr>
<td>Reliability</td>
<td>It is the extent to which the system works without failure.</td>
</tr>
<tr>
<td>Maintainability</td>
<td>It is the ease of maintaining the systems throughout its life cycle.</td>
</tr>
<tr>
<td>Re-usability</td>
<td>It is the extent of reusing several components of the system.</td>
</tr>
<tr>
<td>Portability</td>
<td>It can be the extent to which the system can be moved in different environments or being able to run the software on different platforms.</td>
</tr>
<tr>
<td>Testability</td>
<td>It denotes the extent to which the system under test reacts to the testing process of creating and executing successful tests.</td>
</tr>
<tr>
<td>Interoperability</td>
<td>It is the degree to which the system or the individual components being interconnected for different operations.</td>
</tr>
<tr>
<td>Security</td>
<td>&quot;It is the degree to which malicious harm to a valuable asset is prevented, detected and reacted to.&quot; [14]</td>
</tr>
<tr>
<td>Performance</td>
<td>Different timing characters of the software like Jitter, Throughput, Response Time, etc are referred using this property.</td>
</tr>
<tr>
<td>Quality of Service (QOS)</td>
<td>It is extent to which the system offers its services with good quality being rendered to the user.</td>
</tr>
<tr>
<td>Availability</td>
<td>It is level to which the system is always up for the users to use.</td>
</tr>
<tr>
<td>Scalability</td>
<td>It is level with which the current system can be modified to enhance the current capabilities.</td>
</tr>
</tbody>
</table>

NFR’s cover a wide variety of software quality issues. They are usually referred using the suffixes like "-ilities" or -ities" [8]. The NFR’s are scattered among several categories and are visualized as follows in figure 2.5.
The classification provided in Figure 2.5 has been inspired from [8]. Chung et al. have derived different classification schemes to portray the non-functional system properties. For this study, the author adopted the information from [8] and derived this way of representing the non-functional system properties. They are classified considering different constraints like the Life-cycle, operational, interface, performance and economic constraints. The authors would like to specify that Figure 2.5 is the overview and many other properties can still be broken down into different properties. It cannot be said that this list of properties is not everything. As the complexity of the software systems is increasing, the non-functional properties are expected to diversify.

### 2.3 Related Work

The first publication that the author could find where search-based software testing to non-functional properties has been applied dates back to 1996 [68] with the application of genetic algorithms to test execution time in real time systems. Phil McMinn [49] conducted a survey and reports details about the implementation of search-based testing to NFR’s. Many other researchers continued to contribute to this field where all of them before 2009 have been covered in the systematic review by Afzal et. al [2].

Afzal et al. [2] contributed to this field by conducting a systematic
review focusing on the range of meta-heuristic techniques being applied to test the non-functional properties and the challenges being faced in the application of these techniques. The results of the systematic review show that non-functional search-based software testing has been applied to test properties like Execution Time, Quality of Service, Security, Usability and Safety. Different meta-heuristic techniques used for this purpose are simulated annealing, tabu search, genetic algorithms, ant colony methods, grammatical evolution and swarm intelligence methods [2]. Research prior to this systematic review heavily highlighted execution time attribute where most of them implemented Genetic Algorithm techniques with few limitations/challenges like the non-determinism of the fitness functions, the number of data samples being low, probe effects, etc. Security has also achieved good importance where techniques like Genetic Programming, Linear Genetic Programming (LGP), Particles swarm optimization have been applied. Challenges like maintaining diverse population, additional computation time (ex. LGP) are associated with testing this property. Other techniques like tabu search, hill climbing, simulated annealing have been applied for the usability and safety properties of the system [2]. Combinations of these techniques (ex. TS, SA and GA; SA and HC) have also been applied.

Rim B. Messaoud and Yacine G. Doudane [3] discuss the application of tabu search algorithms to achieve maximum quality of information in mobile applications. Other studies that also focused on improving the quality of service aspect of the software systems are [16] [47] [2] by implementing Multi-objective Genetic Algorithms (MOGA), Particle Swarm Optimization techniques respectively.

Harman et al. [22] give an overall view of search-based software testing in terms of the advantages and the challenges. A deep insight into the non-functional area has not been provided in the article and the authors mention about the lack of research in this field. However, they highlight about a few tools like EvoSuite and Austin which can also be used for the non-functional testing of JAVA and C based software systems respectively.

Most of the studies referred here in this study are either primary or secondary studies that focus on a set of non-functional properties and the application of search-based testing to eventually find out some interesting criteria. Rather than focusing on some properties and trying to find out information like test case creation and execution, the author chose to carry out a study that focuses on the currently used meta-heuristics to test the non-functional properties, list out the challenges being encountered while simultaneously gathering the mitigation strategies from different sources (practitioners / researchers). All this data will be mapped together to form a framework (ex. Guidelines, Checklist) that could be used by practitioners employing NFSBST.
Chapter 3

Method

This section of the document elucidates the research method chosen and the motivation as to why this approach best suits the research conducted. Different sections in this chapter portray the step-by-step execution of several techniques to eventually find out the answers to the research questions. Different empirical research methods exist through which scientific knowledge can be contributed in the field of Software Engineering. The choice of research method has been motivated with reference to the research questions for this study. The choice of the research methodology was based on the context and feasibility of this particular research.

3.1 Research Purpose

The purpose of this study is to compile the information available in the field of Non-functional Search-based Software Testing and present the information in the form of a theoretical framework. It focuses on the different kinds of meta-heuristics being applied to a range of Non-functional properties in different software systems. Also, the challenges identified while employing these techniques and various approaches to solving these problems are investigated. This framework is intended to be used a guide for different practitioners working in this field.

3.1.1 Objectives

Given the overall aim, the main objectives of the research were to:

1. Identify different software systems to which Search-based software testing with respect to the Non-functional system properties has been applied.

2. Analyse different non-functional system properties being optimized or tested. Then find out the different perspectives of the practitioners with regards to these properties.
3. Identify challenges that are usually faced testing these non-functional properties while adopting Search-based testing techniques.

4. Collect different approaches that could potentially solve these challenges and find out the tools used for this purpose.

5. Analyzing the information gathered and finding out a way to map all the information together to form a consolidated framework.

6. Analyzing perspectives of different practitioners and researchers on Non-functional system properties.

7. Validating the framework using the information obtained from interviews.

3.2 Research Questions and Instrument

This research has been inspired from the work of Afzal et al. [2]. To the best of the authors’ knowledge there is no work that aims at getting together all the information and form a theoretical framework. Current literature reports on the ways to implementing Search-based software testing to Non-functional systems properties, challenges that might arise and also state some approaches to solving those problems. Overall, most of the studies focus on the implementation of a single technique to a set of properties and provide deep insight into the actual process of optimizing or testing these properties. This research is aimed at a more generic level and has been inspired by the idea of gathering the information available both in the scientific domain and the industry. To be able to generate the theoretical framework as mentioned above, different bits of information are required. These bits are considered as different research questions and formulated accordingly. As a result, this thesis aims at providing answers for the following research questions and eventually produce functionalities/solutions to this particular problem area.

**RQ1:** What are the different software systems to which Search-based software testing has been applied for?

**RQ2:** What are the different non-functional system properties being tested or optimized using the Search-based testing techniques?

**RQ3:** What are the challenges encountered while employing Search-based techniques to the non-functional properties?

**RQ3.1:** What kind of mitigation strategies / approaches / suggestions exist that could potentially solve the challenges obtained in RQ3?
3.3 Research Method

The research questions stated above incline towards the qualitative part in the problem area making this study overall a qualitative one. Taking a closer look at the research objectives the following interpretation can be made. It is initially necessary to identify the different software systems and various non-functional properties of interest. Various challenges that have been reported in several studies and the corresponding approaches to solve, need to be gathered. Hence, it is appropriate to conduct a Literature review to identify and present the existing body of knowledge related to Non-functional Search-based software testing. The information gathered from the literature review has then been analyzed and an initial version of the framework has been generated. Later on using Semi-structured interviews the information collected thus far has been validated and simultaneously some of the interview questions were open-ended which helped in gathering more information and helped in enhancing the framework.

Figure 3.1: Research Time line

Figure 3.1 is the timeline followed for this research which also depicts how the research questions are answered with the research methods employed for this study.

3.4 Literature Review

The review of past literature and uncovering what is already known about the area of research should never be ignored and it is very crucial for any academic
research [43]. Hart [27] defined a literature review as "the use of ideas in the literature to justify the particular approach to the topic, the selection of methods, and demonstration that this research contributes something new". It is important that the literature reviews include a few characteristics like methodologically analyzing and synthesizing the data and therefore providing a firm base to the research being conducted and then show that the current research contributes to the overall body of knowledge [43]. A literature review, in this case, is used to find out the basic information of the different non-functional system properties being tested using the search-based meta-heuristic testing techniques, the systems to which it is being implemented and the challenges that have been reported till 2015. An initial understanding of the status is provided thereby giving an idea to estimate the extent of information required to be collected using other data collection approaches.

Literature Review for this research has been carried out in different phases as follows. The snowball sampling approach chosen for the literature search, the approach followed to filter literature and then the analysis of the literature are detailed further down in this section. Along with applying snowball sampling as the search approach, a repository of the Search-based Software Engineering research articles has also been inspected to identify even more unique articles [73].

3.4.1 Snowballing as a Search Approach

Snowballing approach has been used for the literature search in this review. This technique as reported in [70] complements the guidelines for conducting literature reviews. It refers to the technique of using both the reference and the citations related to one particular paper and then identifying the additional relevant papers. This approach is not just doing the things as previously done but gives a systematic approach to looking for papers thus reducing the chances of missing out relevant papers.

Why Snowball approach over Database approach?

Many researchers have stressed on the issue of the lack of research for this particular research field [2] [22]. Hence the author chose to opt a more agile approach in finding out the relevant literature rather than the traditional approach of using the database approach. Using the database approach there is a chance of missing out few potentially relevant articles. There could be several reasons for this, where, the most important one is the difficulty in formulating the correct search string using the terminology. If the search string has a general viewpoint, then there are chances of hitting more number of irrelevant papers [70]. A few examples have been described in [70] which show that a few papers have been retrieved using the snowball approach rather than the database approach. When this issue has been investigated the reason was found out to be the inconsistency in the terminology thus affecting the search string. Each approach has its re-
spective advantages and the choice has to be made by carefully looking into the complexity of the research being conducted. In this case, the lack of literature and then for a few papers already retrieved, the concepts are not straight-forward which insisted a deep inspection. Taking these critical issues into consideration snowball approach has been opted which proves to be more beneficial for this study rather than the database approach.

**Database used for finding the Tentative Start set of papers?**

The recommended database for finding the start set of papers for snowball sampling is Google Scholar [70]. The respective advantages of using this database have been specified [70] as it overcomes the problem of publisher bias and access to the papers. There are a few disadvantages as well. The number of records that are retrieved is huge making it difficult to find out the right set of papers. Knisely [38] has mentioned the advantages of using “Engineering Village” as the first choice over “Google Scholar” to search for papers. A study has been conducted by both Knisley Charles and Knisley Karin [38] about finding the information in the most effective way. Databases have been compared about the combination of results being retrieved as a part of the search process. Engineering Village possessed few more additional features than Google Scholar, thus making it a better option. Keeping in mind the drawbacks of using Google Scholar and the advantages of using Engineering Village, the author for this study has made use of the latter to search for the start set of papers as this study required consistency in terminology and the extent to which articles have to be searched. But, Google Scholar has also been effectively used in this process by looking for the full-text of very few articles that the author considered very relevant for this study. In this way Engineering Village has been the primary database and Google Scholar has been used as the secondary database.

**3.4.2 Snowballing Procedure**

As described by Wohlin [70] snowball sampling approach is carried out in four steps. They are outlined as follows:

1. Start Set
2. Iterations
3. Authors
4. Data Extraction

Once the start set of papers to initiate the process are found then the iterations with a combination of backward and forward snowball sampling must be performed to find the additional studies that have been published in the current
research area. Further down, a detailed description is given on how each of these steps are carried out for the work presented in this thesis.

Figure 3.2: Snowballing Process Flow

3.4.2.1 Start Set

Keywords:
To obtain the start set of papers, few activities are implemented. Firstly we need to identify the keywords that hit right on the target. Keywords have been identified observing the research questions. The potential threat of losing important studies in this field has been alleviated with the usage of synonyms for the base keywords. The list of keywords used for the search are tabulated as shown below.

<table>
<thead>
<tr>
<th>Table 3.1: Keywords</th>
</tr>
</thead>
<tbody>
<tr>
<td>Search-based software testing – A1</td>
</tr>
<tr>
<td>Search-based software engineering – A2</td>
</tr>
<tr>
<td>Meta-heuristic testing techniques – C1</td>
</tr>
<tr>
<td>Meta-heuristic techniques – C2</td>
</tr>
<tr>
<td>Meta heuristic – C3</td>
</tr>
</tbody>
</table>

**Search String:**

After the keywords have been identified, the next step is to formulate the search string. It is then used in the repositories/databases to retrieve the articles. To gather the most relevant set of articles from the database, a combination of different Boolean operators like AND, OR and NOT are used in the search string. The search string used for this study is:

\[("A1" OR "A2") OR ("C1" OR "C2" OR "C3")\) AND ("B1" OR "B2" OR "B3")

In snowball sampling approach, it is important to gather a good start set of papers and this is equally challenging with that of the identification of search string for the database search. The author thus carried out the search using Engineering Village (INSPEC) as the database. Earlier to the implementation of the literature review, other search string were used on the databases to check the correctness and the efficiency of the search string used. The search string has been enhanced in several cycles where different terms have been concatenated to the original search string. This was done to obtain all the study pertaining to this research area.

**Tentative Start set:**

The necessary pre-requisites required for snowball sampling are performed and the respective studies have been gathered. The author has chosen to apply inclusion and exclusion criteria created based on the research questions to obtain the most relevant subset for this study [33]. The author decided upon the inclusion of the articles with a thorough inspection on the title and the abstract of the studies available. From this filtered set of studies, the inclusion and exclusion
criteria as described below have been applied.

**Inclusion Criteria:**
- IC1: Articles that are published in English and between years 2009-2015.
- IC2: Peer Reviewed articles.
- IC3: Full-Text availability of the article.
- IC4: Articles that stress on the kind of challenges being faced during the testing process.
- IC5: Articles focusing on the testing of non-functional properties of the software systems.

**Exclusion Criteria:**
- EC1: Studies that do not directly link the specific contextual factor with testing non-functional requirements using search-based testing techniques.
- EC2: Studies that are an earlier version of the latest study done in the current field.
- EC3: Review Papers; Only the references of the review papers were examined to find most of the primary study done in this research area.

The inclusion and the exclusion criteria were applied to the set of articles that were retrieved from the database search. As depicted in figure 3.3, the search string execution in the database has resulted in 62 articles in total. Upon the application of the first inclusion criteria, the set has been filtered to 54.
Chapter 3. Method

There is a systematic review [2] published in 2009 in the field that is currently being researched. Hence for this literature review, we have considered only those articles published after this systematic review. After this phase, the inclusion criteria IC3 and exclusion criteria EC2 have been applied which further filtered the article set to 51. There was a systematic mapping study [1] which is the earlier version of [2], has been excluded considering EC2 and two articles had no full-text availability. Then, these 51 articles have been subjected to Title inspection and this phase resulted in 29 unique articles. Then the abstracts of these articles have been thoroughly inspected, resulting in 18 articles. Next, a full-text review has been performed on these articles. Full-text review in this context means that the whole paper is browsed looking into critical areas like the introduction, discussions, conclusions, etc. This is a crucial phase in the whole process because the steps after this largely depend on the relevance of the articles. This phase resulted in 14 articles which have been examined by the author and the supervisors’ support where another article has been eliminated from the set finally resulting with a start set of 13 unique articles.

Following the approach detailed in [70], the next step in the snowball approach is to carry out several iterations of the Backward and Forward searches to find relevant articles that could have been missed using the database approach. Performing this approach will help in finding more articles that are relevant for this study which further helps in gathering more information.

3.4.2.2 Snowball Iterations

Once the start set is finalized, the next step is to conduct the iterations of backward and forward snowball sampling. A brief description of both these techniques is given below.

Backward and Forward Search:

**Backward Snowballing:**
Here the reference of a particular paper is used to identify the new papers that could be included for the literature review. For this research, the inclusion of the papers has been done following the criteria listed below.

1. **Publication Venue:** There are chances of finding few additional papers by searching for the publication venue where relevant papers are previously published. Another reason for this search is that the author needs to make sure that the papers published before 2009 must be ignored because they would be covered in the systematic review mentioned above.

2. **Title:** First the titles of the list of references is thoroughly inspected. Inspection of the title involved looking for specific keywords which are close to the keywords that have been used during the search string formulation.
This helped in eliminating those articles that are completely irrelevant for this research. Also, other basic criteria like the language have also been examined to filter the papers.

3. **Abstract:** After the title and other basic criteria have been examined for all the articles, in the next level of searching for relevant articles in backward snowball sampling the abstracts of those articles whose title seems to be tentatively relevant have been examined.

4. **Author:** For this research two authors have contributed very actively to most of the research in Search-based Software testing. This also helped the author in looking for their work and be able to go through a few additional papers and check for their relevance.

**Forward Snowballing:**

This refers to inspecting the papers that cite the current paper being examined. INSPEC being the prime database for this study, the author initially chose to look for citations in this database. After careful inspection, it is found out that the set of articles retrieved from INSPEC is a subset of the set of articles retrieved using Google Scholar. Hence the author chose to use Google Scholar for the purpose of finding the citations for the papers. Only citations are used and quotes are removed [70].

The same approach followed for backward snowball sampling is implemented for this in finding the relevant articles. The title is examined and then the abstract is read. If insufficient information is found then the full-text is read.

The list of articles found as a result of the first iteration are further used for the second iteration. Papers found through these iterations are only used for the analysis. For this research two iterations have been implemented and 18 relevant articles have been found in total.

### 3.5 Validating the Framework

The findings of the literature review have been used to form the initial version of the framework. Hence, the one developed thus far has to be validated. The details of the information obtained is available in ???. Several ways exist through which the information can be validated. It can be done through conducting semi-structured interviews, or conducting a survey from a target population. For the work presented in this thesis, the author chose to conduct semi-structured interviews as a way to both validate the existing information and also enhance the existing framework. With semi-structured interviews, both the author and interviewees have the freedom to communicate and exchange valuable information at ease. The author can try to extract most out of an interviewee which by conducting a survey could be a limitation. Moreover, the framework generated thus
far lacks information. As a result of conducting these semi-structured interviews the author expects to gather information on a more generic level and expand the framework which consists of data focusing on more number of non-functional system properties.

### 3.6 Semi-Structured Interviews

In general, there are three kinds of interviews that anyone could put to practice. They are Structured, Semi-structured and Unstructured Interviews [9].

- **Structured Interviews**: These kinds of interviews are mostly close-ended. Here, the respondents are more likely to give known answers.

- **Semi-structured Interviews**: The instrument used in these kinds of interviews consists of both open and close-ended questions thus giving the freedom to both the participants to express their knowledge regarding the subject to a fairly decent extent.

- **Unstructured Interviews**: These kinds of interviews do not necessarily follow a protocol. Much of the control lies in the interviewee and the interviewer has very less scope of retrieving relevant information.

The Semi-structured interviews conducted for this research are used to complement, validate the results of the literature review and also used to further enhance them. From the interviews, the author intends to gather other approaches/suggestions to solve the challenges that have been collected till now and at the same time, identify new ones. For this thesis, collecting data is more feasible through conducting interviews rather than applying quantitative measures. Moreover, through interviews, people provide more insight into their world, the opinions they have towards a particular aspect [28]. The author chose to use Semi-structured interviews as they often give both the interviewer and the interviewee to deviate from the protocol and collect data based on the opinion, memory or interest [61]. A completely unstructured interview has the risk of not being able to elicit the required information and sometimes even incline towards a different topic or theme altogether. Structured interviews mostly contain close ended questions to which the respondents are more likely to give known answers. A balance of open and close-ended questions has been maintained therefore allowing the author to explore and improvise the content available in this subject.

#### 3.6.1 Interview Subjects

The interview subjects were selected considering the following criteria. The resources available, the experience of the interviewee, relevance to the sub-
subject being investigated and the access to the information the author needs are considered while selecting the interview subjects. Finding relevant practitioners implementing Search-based testing techniques to Non-functional properties has become problematic as this field is relatively new. According to one of the interviewee this subject is still inclined more towards research than being used in the industry. Hence, the author tried to find academic researchers who have relevant experience in Non-functional Search-based software testing and could provide further information regarding this subject. With the intention of validating the work, a second set of interviewees were selected, they are practitioners who have a strong background in Non-functional testing. The selected interviewees were invited by email following a common email template containing a short description of the purpose of the study. A total of 9 interviews were conducted for this study. Three subjects with a strong background in Non-functional Search-based software testing and 6 practitioners working in Non-functional testing domain have been interviewed for this research. Convenience sampling has been used for the selection of interview subjects. Table 3.2 contains the detailed list of interviewees. The experience mentioned in the table below represents the experience of the participants in Non-functional testing using Search-based testing techniques or other testing strategies (researchers and practitioners). One of the interviewee has been working on Non-functional properties for three years but he has been working on Search-based software testing for over a decade.

Table 3.2: Interviewee Details (Demographics)

<table>
<thead>
<tr>
<th>ID</th>
<th>Role</th>
<th>Experience in Non-functional testing (years)</th>
<th>Interview duration (minutes)</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT1</td>
<td>Researcher in Search-based software testing</td>
<td>8</td>
<td>60</td>
</tr>
<tr>
<td>INT2</td>
<td>Researcher in Search-based software testing</td>
<td>10</td>
<td>64</td>
</tr>
<tr>
<td>INT3</td>
<td>Researcher in Search-based software testing</td>
<td>3</td>
<td>Email Interview (Not Applicable)</td>
</tr>
<tr>
<td>INT4</td>
<td>Test and Deployment Analyst</td>
<td>5</td>
<td>45</td>
</tr>
<tr>
<td>INT5</td>
<td>Test Consultant</td>
<td>10</td>
<td>52</td>
</tr>
<tr>
<td>INT6</td>
<td>Test Lead and Manager</td>
<td>5</td>
<td>67</td>
</tr>
<tr>
<td>INT7</td>
<td>Senior Software Test Engineer</td>
<td>2</td>
<td>32</td>
</tr>
<tr>
<td>INT8</td>
<td>Developer and Tester</td>
<td>3</td>
<td>42</td>
</tr>
<tr>
<td>INT9</td>
<td>Practice Head, QA and Testing</td>
<td>12</td>
<td>77</td>
</tr>
</tbody>
</table>
3.6.2 Interview Design

The interviews were conducted face-to-face, via Skype and Emails. The questions were both open-ended and close-ended, fundamentally focusing on capturing the information related to their testing approach with relation to Non-functional system properties, issues or challenges they face and their approaches to solve a few problems. Time glass approach as described in [61] has been followed where the interview starts with a set of open-ended questions (for examples, the participant’s demographics), moves towards more specific questions and towards the end concentrates again focuses on a few open-ended questions. One full hour has been requested from each participant taking into consideration the nature of the instrument being used and the study being conducted. One of the interview was carried out using Emails. The same interview instrument has been used to this participant as well. Also, a tiny description next to the questions consisting of the authors intention is also specified just to make sure that information is accurate and the difference in understanding / interpretation is eliminated.

3.6.3 Interview Instrument

The interview instrument used for this research has been formulated over a series of iterations and on regular interaction with the supervisor. Experiences on how to conduct semi-structured interviews [28] came in handy while framing the questions. Open-ended and close-ended questions were framed keeping in mind the research objectives and the data collected from the literature review. The questions have been framed in its simplest terms to make sure that they are understood easily. Furthermore, the data collected from the literature review lead to a few interesting points to be discussed and clarified from these interviews. The initial interview instrument contained questions on a general perspective with the main intention of gathering information on knowing the non-functional testing approaches being followed.

Next the interview instrument has been discussed with the supervisor to find out any flaws or any irrelevant questions. The author feels that this is one of the most important thing to be done before the instrument is used practically on the interviewees. It is because with such conversation interesting points can be found out and eventually improving the quality of the instrument. This process has been done in a few iterations. The final version of the instrument has had a significant difference where the questions have been categorized into three tiers. Tier 1 questions included the demographic questions, Tier 2 questions concentrated more on the approach the interviewee followed and the challenges he/she faces regularly while testing the non-functional system properties and Tier 3 questions concentrated specifically on the Search-based testing to non-functional system properties and validating the information from the literature review.
As suggested by Yin [71], the author felt it is necessary to conduct a pilot study. Following the suggestions, during the pilot interview I focused on criteria like the interview session - if it is too long, and interview instrument - if the questions are being understood clearly and most importantly if the questions are relevant and if they are of right detail. Feedback from the pilot has been used to make minor tweaks to the interview instrument.

The interviews lasted between 30-75 minutes. All the interviews have been audio-taped and this has been done only after the interviewee has accepted to it. Apart from that the author also took few notes on some important details that he thought would be necessary for answering the research questions. Audio-taping has become a common practice and has proved to be an effective way of capturing the information [28].

After each interview, extensive notes were taken listening to the recording and in a few instances the interviewee has been contacted for further clarification. Extensive notes are different from full-transcripts where the focus is kept on identifying important quotes or examples the interviewee’s state [59].

### 3.7 Data Analysis

Conducting semi-structured interviews leads to the collection of large amounts of data. To properly analyze the qualitative data, the aid of a suitable approach is necessary [62]. Yin [71] stresses the importance of maintaining a clear chain of evidence right from the collection of data till reasonable conclusions are made on the subject. A clear and precise approach has been presented below.

Different approaches like Content analysis, Grounded Theory exist to analyze the qualitative data, but everything depends on the basics of coding and finding patterns [59]. For this research the author chose thematic analysis as described by Braun and Clarke [5]. Thematic analysis provides a way to identify important themes from a large data set. Following the approach as described in [5], there are six steps and they are represented in figure 3.4.
Step 1: **Familiarize yourself with the data** - The main goal of getting familiarized with the data is that it becomes easy to identify the interesting things that need to be highlighted as a result of the analysis process. With repeated reading of the data, search for meaningful data, patterns, etc to be made. The recorded interviews were transcribed into a document using the tool ExpressScribe. The field notes and the extensive notes were also referred while the transcription is being done to identify any abnormalities and check the accuracy. The transcripts were thoroughly examined during and after the process.

Step 2: **Generating initial codes** - This phase involves the creation of an initial set of codes from the data set. The document has been imported to a tool named Nvivo, a qualitative analysis tool offering organization and visualization of the data. Both open and close coding have been used for categorizing the data. Open coding refers to the process of identifying codes from the transcripts and building concepts out of those codes. Close coding refers to the process of identifying the code prior to looking at the data transcripts. A set of codes are formed relevant to the subject being inspected. The field notes and the extensive study that has been done supported in finding out the categories. Close coding has also been used to form an initial set of codes looking at the research questions. For example, RQ1 is about the different software systems to which Non-functional SBST has been applied for. Hence, codes like, system, software systems, etc have been formulated prior to the analysis. Similarly the same has

---

been applied for all the research questions. For a clear view of how the codes have been used in the tool refer to figure 3.5.

**Step 3: Searching for themes** - In this phase classification of the codes into different themes is done. Themes are something that capture an important aspect of the data that is relevant to the research question [5]. According to Braun and Clarke, there is no hard-and-fast rule to determine the level of correlation between the theme and the research questions. The themes can be self-explanatory and should contain relevance with the research questions. Here the themes are classified as per the research questions and the corresponding sub-themes have also been divided. Refer to figure 3.5 for a better understanding.

**Step 4: Reviewing themes** - The themes generated so far have been carefully reviewed by the author. The relevance of the themes has been inspected by comparing it with the data set. Recurrence of the patterns in the data, arguments made by the interviewees have been considered. A few changes have been made to the themes generated, where some have been added, some removed and others merged. These themes are then used for generating the thematic maps.

**Step 5: Defining and naming themes** - This is an important phase in the analysis process. Sometimes choosing a name to the themes can be very tricky and this has to be done carefully. The final theme name shown in figure 3.5 has either been generated in the first attempt or after several changes until it is meaningful. The names of these themes have been framed in the most general terms.

**Step 6: Producing the report** - In this final phase of the analysis process, a clear, concise and logical report of the results has to be prepared. The reader has to be convinced with the final outcome and the whole motive of this research [5]. The results of the interviews have been presented in chapter 5.
Figure 3.5: Themes formed using Nvivo
Chapter 4

Results of Literature Review

The present study revolves around the concepts of non-functional system properties, testing and optimizing these properties using search-based techniques, different kinds of meta-heuristic techniques being used, various challenges being faced both in academia and industry, tools that are specifically used for this purpose. Before we dive deep into the study, an initial understanding of the concepts surrounding this research has been provided, systems under test using SBST, the history of Non-functional Search-based Software Testing (NFSBST), challenges being encountered and the need for a theoretical framework guiding the practitioners and SBST researchers who would be implementing this process to optimize the software systems.

A systematic review [2] very close to the current research has been conducted and all the information prior to this systematic review has been gathered from there. A total of 25 articles have been analyzed for this literature review whose details have been provided in Appendix D.

It is already stated in chapter 3 that all the research questions would be answered using the literature review and then further data collection would be done through conducting interviews to validate the existing information. Each of the research questions contains a dependent variable for this research. Hence the following sections are categorized according to these dependent variables where the literature addressing each of these dependent and independent variables is applied, analyzed, synthesized and evaluated. Google Fusion Tables 1 have been used to traverse through the data collected from the literature review and present it.

4.1 Software systems under test

In this section we identify different kinds of software systems that have undergone the testing or optimization processes using Search-based software testing. A system under test (SUT) is that which is being tested for the correctness of the operations it is bound to perform. SBST has been applied to systems ranging

---

1Available at: https://support.google.com/fusiontables/answer/2571232
from Mobile applications to Component-based systems in Automotive domain. Reviewing all the articles the different kinds of software systems identified have been tabulated below. The information presented in the table below has been extracted from sources [3, 6, 12, 20, 32, 41, 46, 47, 51, 54, 56, 69].

<table>
<thead>
<tr>
<th>S.No</th>
<th>System Under Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mobile Applications [3]</td>
</tr>
<tr>
<td>2</td>
<td>Individual Algorithms of the software system [32]</td>
</tr>
<tr>
<td>3</td>
<td>Randomised Algorithms [54]</td>
</tr>
<tr>
<td>4</td>
<td>Micro-Controllers (Embedded Systems) [56]</td>
</tr>
<tr>
<td>5</td>
<td>MiniSAT Boolean Satisfiability solver [6]</td>
</tr>
<tr>
<td>6</td>
<td>Software systems developed using C-language [69]</td>
</tr>
<tr>
<td>7</td>
<td>Software systems developed using JAVA (Object Oriented Programming)</td>
</tr>
<tr>
<td>8</td>
<td>Software systems developed using C++ [41]</td>
</tr>
<tr>
<td>9</td>
<td>Cloud systems [46]</td>
</tr>
<tr>
<td>10</td>
<td>Safety Critical systems [12]</td>
</tr>
<tr>
<td>11</td>
<td>Service Oriented systems [20, 47]</td>
</tr>
<tr>
<td>12</td>
<td>Component based systems in Automotive domain [51]</td>
</tr>
</tbody>
</table>

Some of the systems under test identified here are self-explanatory while some reveal very less information. Here, some of the SUT that require an insight have been described briefly. "Individual algorithms of the software" (SUT 2) here resembles the functions that are used in the development of the software. As discussed earlier, the application of SBST can be scaled to software of any size or complexity. Randomised algorithms (SUT 3) are those which make stochastic decisions based on the results obtained from the probability distributions. For example, toss of a coin [54]. MiniSAT Boolean Satisfiability solver (SUT 5), is a satisfiability solver which determines if there exists an interpretation that satisfies the given Boolean formula. The context in which this SUT has been involved here is when Harman et al. [6] try to derive an energy efficient solver. For SUT 6 resembling software system developed using C-language is where the authors take a simple function written in C and try to evaluate the semantic correctness (equivalence) of the program [69]. SUT 12 which resembles Component based systems in automotive domain represent the components of a system. Meedeniya et al. [51] represent component as a system element, for example, an Electronic Control Unit (ECU). ECU is a computer in itself and has a software installed in it.
Chapter 4. Results of Literature Review

4.2 Meta-heuristics

This section deals with the different meta-heuristics that have been implemented as a part of using SBST to optimize the non-functional system properties. Most of the researches analyzed here implemented only one technique to solve a particular problem except for a few which have made use of a combination of techniques which will be described in detail in the sections below. Some of the algorithms have been used by multiple researchers and the most popularly used is the Genetic Programming and its versions like the Genetic Evolution, Genetic Algorithms and Genetic Improvement. The information presented in the following table has been extracted from the following sources [2,3,12,16,20,21,24,26,41,46,47,51,54,69,72].

Table 4.2: Meta-heuristic Techniques

<table>
<thead>
<tr>
<th>S.No</th>
<th>Meta-heuristic Techniques</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Tabu Search [2,3,46]</td>
</tr>
<tr>
<td>2</td>
<td>Multi-objective Genetic Algorithms (MOGA) [16]</td>
</tr>
<tr>
<td>3</td>
<td>Amortised Optimization [72]</td>
</tr>
<tr>
<td>4</td>
<td>Genetic Algorithms [2,3,20,41]</td>
</tr>
<tr>
<td>5</td>
<td>Simulated Annealing [2,46]</td>
</tr>
<tr>
<td>6</td>
<td>Hill Climbing [12]</td>
</tr>
<tr>
<td>7</td>
<td>Genetic Programming [24,56,69]</td>
</tr>
<tr>
<td>8</td>
<td>Particle Swarm Optimization [47]</td>
</tr>
<tr>
<td>9</td>
<td>Genetic Evolution</td>
</tr>
<tr>
<td>10</td>
<td>Multi-objective Optimization [16,24,51,69]</td>
</tr>
<tr>
<td>11</td>
<td>Genetic Improvement [6,21,26,54]</td>
</tr>
<tr>
<td>12</td>
<td>Ant Colony Optimization algorithm [51]</td>
</tr>
</tbody>
</table>

4.3 Non-functional system properties

This section discusses about the different non-functional systems properties that have been addressed by several researchers. It has been identified that the set of non-functional properties being addressed in any problem deeply depends on the type of system being examined. Not just one property is tested for a system but instead a set of them are tested depending on the importance and few other crucial factors. In the following table the list of properties reported in the literature have been tabulated. The list of properties tabulated below have been identified from the following literature sources [2,3,6,12,16,20,24,26,40,45–47,51,54,56,72].
Table 4.3: Non-functional properties reported in the literature

<table>
<thead>
<tr>
<th>S.No</th>
<th>Non-functional system properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Quality of Information [3]</td>
</tr>
<tr>
<td>2</td>
<td>Reliability [16, 46, 47, 51]</td>
</tr>
<tr>
<td>3</td>
<td>Performance [12, 20, 32, 46, 72]</td>
</tr>
<tr>
<td>4</td>
<td>Cost [45, 46, 51]</td>
</tr>
<tr>
<td>5</td>
<td>Availability [45–47]</td>
</tr>
<tr>
<td>6</td>
<td>Response time [24, 51]</td>
</tr>
<tr>
<td>7</td>
<td>Throughput [24]</td>
</tr>
<tr>
<td>8</td>
<td>Memory Consumption [21, 24]</td>
</tr>
<tr>
<td>9</td>
<td>Security [2]</td>
</tr>
<tr>
<td>10</td>
<td>Safety [2, 45]</td>
</tr>
<tr>
<td>11</td>
<td>Efficiency [40]</td>
</tr>
<tr>
<td>12</td>
<td>Self-stabilization time [54]</td>
</tr>
<tr>
<td>13</td>
<td>Power/Energy Consumption [6, 26, 45, 56]</td>
</tr>
<tr>
<td>14</td>
<td>Service Time [54]</td>
</tr>
<tr>
<td>15</td>
<td>Processor Utilization [45]</td>
</tr>
<tr>
<td>16</td>
<td>Usability [2]</td>
</tr>
<tr>
<td>17</td>
<td>Execution time [2, 41, 47, 56, 69]</td>
</tr>
</tbody>
</table>

Table 4.3 contains the list of properties. From the findings of the literature review, it is identified that the most frequently addressed attributes are Performance, Execution time and Power/Energy consumption. All of them come under the category of non-functional system properties but the author observed that properties like Efficiency, Robustness, Throughput and Reliability for some SUT become the sub-aspects of Performance.

4.4 Challenges encountered

Optimising the programs for non-functional properties such as those mentioned above can be very demanding. It is equally difficult for the programmer/developer to address all these at once in the source code [24]. Potential challenges that have been reported by various authors have been listed below.

4.4.0.1 Challenge 1 - Conflicting Properties (GISMOE)

Mark Harman et al. in [24] state that it is difficult to propose just a single solution to a problem because the non-functional properties are often conflicting with each other. The authors use a framework called GISMOE (Genetic Improvement of Software for Multi Objective Exploration). It is a form of automated testing which assesses the degree to which the functional properties are
preserved by the software engineer and the how the non-functional properties are achieved by the machine. An example to explain this challenge would be, when attempts are made to reduce the response time of the program, effect on other quality attributes like power, memory, etc. is inversely proportional. Hence multiple solutions might exist to a problem and these have to be sought out too.

**4.4.0.2 Challenge 2 - Fitness function measurement (GISMOE)**

The authors while proposing the GISMOE [24], an approach that distinguishes the functional and non-functional aspects of the system, mention a few challenges. One of the problem is "Measuring the Non-functional properties as a fitness function". It is already known that generating the fitness function is one of the core ingredients to implement search-based software testing. The main challenges arise when trying to interpret these complex properties as fitness function. For these properties to be optimized or tested it is very important for them to be measured. By measurement, the authors mean that they have to be sufficiently precise and well defined. This is an important requirement of GISMOE to work.

**4.4.0.3 Challenge 3 - Test Data Generation (GISMOE)**

Another challenge the authors mentioned is during the "Test Data Generation". Analyzing the programs from different perspectives is crucial to consider test data generated for all the properties rather than depending on the tests generated for the functional properties. This poses a challenge as there is a requirement for fast generation of the test cases for a rapidly changing suite of inter-related programs.

**Approaches to Solve:** Search-based software testing targets the non-functional system properties and GISMOE makes the process of test generation completely automated thereby allowing greater confidence in the code generated. GISMOE does this by adapting different techniques and then develops test cases on-the-fly [24].

**4.4.0.4 Challenge 4 - Balancing exploration and exploitation**

Simone A. Ludwig in [47] applies two meta-heuristic methods based on Particle Swarm optimization for optimizing quality attributes like performance, reliability, security, etc. One of those methods is the Hybrid-Particle Swarm optimization and main challenge while applying these methods is to find a right balance between the behavior of the algorithm while searching the search space (exploration) and the right search to find the global optimum (exploitation).
Chapter 4. Results of Literature Review

**Approaches to Solve:** Using hybrid approaches can solve this issue. Different studies which show successful implementation of this approach. In [53] a combination of PSO and hill climbing helps in improving the performance.

4.4.0.5 Challenge 5 - Sensitivity to Environment

David R. White in [69] mentions that the primary concerns while testing a software mainly lies in the programming language, target platform, compiler and meeting non-functional requirements alongside is a tough task to achieve. Even small changes to the source code leads to very unusual behavioral changes to the programs execution time. These properties being interdependent with each other present a few trade-offs to the programmers.

4.4.0.6 Challenge 6 - Maintaining semantic equivalence

The proposed framework in [69] takes code written in C language as the input and applies Genetic programming to optimize one or more non-functional attributes at the same time. Maintaining the semantic equivalence in the code during this process is the biggest challenge. With semantic equivalence, the authors mean that two programs are equivalent if and only if the same behavior is being shown by them for different input satisfying the given condition. The authors further mention that this framework could provide a deep insight into the potential optimizations possible but does not provide a full automated approach to optimization. They feel that the output might not be semantically equivalent to the input function and hence the results must be verified manually.

**Approaches to Solve:** To gain confidence in the results and evaluate the correctness of the output Coevolution algorithms can be used to generate coevolved test cases and test the semantics of the program [69].

4.4.0.7 Challenge 7 - Connecting Energy consumption and program source code

When it comes to optimizing the energy consumption of a specific system, Bobby R. Bruce et al. [6] consider the connection between the source code written and the energy consumed by the compiled project as one of the largest hurdles. Without a deep understanding of how the compiler works and the energy consumption for a specific task, this problem continues to be a nightmare to the developers and testers. Capturing the energy consumed outside of the CPU can be an equally challenging task.

4.4.0.8 Challenge 8 - GIGI4 (Factors affecting energy consumption)

Another study on optimizing energy consumption using GIGI4 framework by Mark Harman and Justyna Petke [26] state that finding the other factors
that affect the energy consumption and their relative contribution is a challenge. However, the authors also state a mitigation strategy to this problem.

**Approaches to Solve:** The above-mentioned challenge could be addressed by using Genetic Programming. The energy consumption equation has to be formulated inclusive of the parameters that determine the ways in which devices, memory, and screen are used. It is also mentioned that generating the equation from scratch is not necessary and several candidate equations exist. But improving this candidate equation for better results is a different problem altogether [26].

### 4.4.0.9 Challenge 9 - Improvising Search space

Genetic Improvement a variant of Genetic Programming has been specifically used for exploring the search space of several program variants [21]. The process of optimization/testing involves adjustment of an already functioning software. Saemundur O. Haraldsson and John R. Woodward mention that this process is highly susceptible to compiler errors eventually resulting in non-terminating programs [21].

**Approaches to Solve:** Anticipating this issue and overcoming this could be possible by involving a penalty and stopping the program to enter the next iteration. Another method involves eliminating the population from the search space of the program being evaluated [21].

### 4.4.0.10 Challenge 10 - Efficient fitness function

Whenever the programs are optimized or tested, they are done against a set of objectives that are set by different stakeholders. Finding an efficient fitness function to different objectives can be very challenging while maintaining the exact functionality of the system [21].

### 4.4.0.11 Challenge 11 - Faulty test cases

While applying Genetic Improvement process the probability of faulty individuals out of scope, semantically incorrect test cases is very high [21]. The authors cite this information in a generic context.

### 4.4.0.12 Challenge 12 - Automotive redundancy allocation impact

Indika Meedeniya et al. [51] conduct a case study in the automotive domain where they try to find out the impact of redundancy allocation on different non-functional attributes like reliability, response-time, etc while applying Multi-objective ant colony optimization. They found out that while applying redundant allocation to improve the reliability of the system, there can be a negative impact on the other non-functional attributes. Additional overheads incur into the
response-time of the system which itself is another important attributes of the automobiles.

All the challenges mentioned above just state the problem irrespective of the system being tested and the complete set of non-functional attributes being tested and the meta-heuristic that has been applied. Further down in this chapter, a diagrammatic representation has been provided to get a better understanding and to give this framework a shape which would be further enhanced by the data collected through the interviews.

4.5 Tools Used

In this section the list of tools that have been used for the purpose of testing the non-functional attributes of the system is presented. Not many tools used have been reported in the literature. Some of them are used for a wider scale of programs while some of them are specifically used for a particular problem. The list of tools the attributes being addressed is mentioned in Table 4.4.

<table>
<thead>
<tr>
<th>S.No</th>
<th>Tool</th>
<th>Domain of usage</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Performance-Enhancing Tool using UML Transformations and Multi-Objective Optimisations (PETUT-MOO)</td>
<td>Architecture Optimisation</td>
</tr>
<tr>
<td>2</td>
<td>EvoChecker</td>
<td>Quality of Service (QOS) properties</td>
</tr>
<tr>
<td>3</td>
<td>MOS tool</td>
<td>Safety properties</td>
</tr>
<tr>
<td>4</td>
<td>EVOSUITE</td>
<td>Used for Java based programs</td>
</tr>
<tr>
<td>5</td>
<td>Augmented Search-based Testing (AUSTIN)</td>
<td>Used for C based programs</td>
</tr>
</tbody>
</table>

Table 4.4: Tools Used
Chapter 5

Results and Analysis of Semi-structured Interviews

This chapter presents the results of the Semi-structured interviews conducted for this study. The data collected has been analyzed using Thematic analysis, [5] described in section 3.7.

Information in this section has been categorized into two sections namely, Search-based testing domain and Non-functional testing domain. For this thesis, two sets of interviews were conducted. The first set of interviews were three, who are academics/researchers that work with developing and validating SBST techniques and the other set of interviews were six, who are the practitioners that have relevant experience in Non-functional software testing. Both groups have the relevant experience, just that no interviewee has a complete picture of the phenomenon being investigated, the two groups have been interviewed. Interviewees with relevant experience in this subject (Search-based software testing) are referred to as INT1, INT2 and INT3 (interviewee ID). Remaining interviewees (experienced in Non-functional testing) are referred to as INT4, INT5 and so on till INT9. As mentioned earlier the sampling method is convenience sampling. The information that has been collected from the researchers in Search-based software testing has only been used to enhance the framework. The information from the other set of interviewees has been used to validate the findings and also to find out suggestions on a generic level.

After reporting the new answers found to the research questions, in the latter part of the results, data related to information validation (section 3.5) has been reported. Diagrammatic representation of the information gathered has been presented below. We will refer to these figures during the individual analysis of the research questions.

The figures figure 5.1 and figure 5.2 contain the information furnished by the interviewees. The interview transcripts have been processed using the thematic analysis technique and the data has been broken down according to the research questions. The second level nodes of the tree contains the interviewee ID and the next level nodes represent the system(s) they are currently working on or have worked on in the past. Next, the core non-functional properties of interest have been listed. Then there is the list of challenges they think that could affect
the process of testing or optimizing these properties. Finally, the approaches to solve a few of these issues have also been listed. However, a correlation between the challenges and the suggestions has not been depicted in the tree diagram. The respective details can be found in the individual analysis sections described further down in this chapter.

Figure 5.1: Researchers in Search-based testing domain
Chapter 5. Results and Analysis of Semi-structured Interviews

5.1 RQ1 - Software systems under test

This section answers the first research question related to different software systems under test to which Search-based testing techniques are being applied for. Identified system under test (SUT) have been shown in the form of a thematic map in figure 5.3.
It can be observed that the system under test theme is classified again into two sub-themes. They are, "NFSBST", which resembles those system under test which are being optimized in the Search-based testing domain and "Regular Non-functional testing", which resembles those system under test undergoing Non-functional optimization. Each of the interviewees who has expertise in SBST has been working on different software systems and there is not much overlap. One of the researchers works on producing tools and techniques that could be used over a range of systems to change the way the fundamental aspects of the software systems do. The source count for this theme is 3. When it comes to Regular Non-functional testing the source count is 6 with few practitioners working on a range of systems. It has been observed that 5 out of the 6 practitioners work on the web-applications or web-portals and other systems include mobile applications, stand-alone applications and RFID technologies in embedded systems.

5.2 RQ2 - Non-functional properties tested

This section answers the question of what kind of Non-functional properties are of prime concern when it comes to the application of Search-based software testing techniques. Various perspectives related to non-functional requirements have been gathered from the interviews. During the course of interviews, the author has noticed that the way Non-functional properties are interpreted influences the approach in testing them. Looking at the data available a different dimension to how these properties are interpreted has been observed. Non-functional properties of interest largely depend on the resources available, the system under test and several other factors. Further down information given by different interviewees on their perception of Non-functional properties, the factors which
Chapter 5. Results and Analysis of Semi-structured Interviews

influence the selection of NFP’s, prime NFP’s of interest and which of them are being given the most importance has been described. The respective thematic map is presented in figure 5.4. Non-functional properties being tested is the main theme with three different dimensions to answering this using the interviews. First sub-theme is the Non-functional property interpretation where the author tries to understand how the properties are being interpreted by various practitioners. Next sub-theme is Non-functional properties of interest which identifies the various aspects of the systems that are being targeted in the corresponding SUT. There are three different sub-themes here which resemble the factors affecting the selection of those aspects. They are System under test, resources available and stakeholders perspective. Stakeholders perspective in this context means that different people like Customer/Product Owner, Project Manager, etc. can also influence the selection of the properties. For example, a customer would like his web-application to look very pleasant and be very easy to navigate through it. Therefore this leads to concentrate on the usability aspect of the system. Finally, the last sub theme is regarding the importance of a certain set of non-functional aspects.

Figure 5.4: Thematic map of Non-functional requirements tested using SBST

5.2.1 Interpreting Non-functional properties

After careful inspection of different perceptions given by the interviewees, all of them relate non-functional properties to the functional properties. Non-functional properties more or less relate to the functional properties in terms of its quality or some additional value, etc. A few interesting interpretations gathered from the interviewees are listed below:

According to INT1 "[Interpretation of Non-functional properties] depends on the
system under test. For some systems, it can be a functional property in the sense that it something that needs to be built. For example - Safety is one such property."

INT2 talks that "The interpretation is much about communicating with the people rather than a hard and fast definition to it..."

INT3 perceives that "Non-functional properties are aspects of the system that cannot be specified as a formal semantic."

According to INT9 "[The non-functional properties] are not tangible or quantifiable to the end user but they are something that adds an extra edge to the quality of experience."

Other interpretations are more or less the same where they stress either on the quality of the system or enhancing the functional aspects of the system. Apart from the textbook definition stated earlier in this document, these are a few thoughts on how Non-functional properties are being interpreted.

5.2.2 Non-functional properties of interest

When asked about what kind of properties are of main concern for the applications being worked on by the interviewees the following impressions have been made.

Few interesting criteria have been identified regarding the selection of properties. Targeting a set of Non-functional aspects for a particular system depends on several factors. According to INT1 "It is not always possible to target every non-functional aspect of the system", wherein he tries to specify the point that factors like the SUT, domain, influence the selection of a set of properties. All the interview participants accept that the SUT deeply influences the selection of Non-functional properties. Our prime concern here being the application of Search-based testing techniques, the following results have been obtained. INT1 with relevant Search-based testing experience is working on the Performance and Testability properties of the embedded systems. INT2 in the same domain (Search-based software testing) is working on the same set of properties (Performance, Reliability) except that it is only Performance in this case. But the difference is the kind of techniques being applied. INT1 applies Genetic Programming to form the search whereas INT2 uses hill climbing and Monte-Carlo tree search to optimize the properties. INT3 is working on the wall-clock execution time on Linux systems. INT3 is trying to find out ways to use program change history that can aid in automatic debugging of these systems.

Looking at the results obtained from the practitioners in Non-functional testing the following analyses has been made. Majority of the practitioners work on web-applications along with some other applications like RFID technology in embedded systems, mobile applications, etc. The most important non-functional aspect to all of them is Performance. As told by one of the interviewee "Per-
formance is of more value for us in terms of business". But when asked what apart from performance do you think is important, the interviewee opted for Usability. Three practitioners out of 6 think that Usability is the next important non-functional aspect. It is also found out from the interviewees that there is a big open question of how automation is done to test the usability of a product. Others include the Security aspects of mobile applications, Throughput, Stability and robustness which are different attributes under Performance.

To understand why Performance is of most importance to both the people working in Search-based testing and practitioners in Non-functional testing the following is the summarized opinion "There exists a lot of difference between simulation and real-time execution. Things are never the same and hence more effort has to be put into this." The findings of the literature review motivated the author in discussing another interesting criterion of most of the research done on either performance or execution time or energy consumption aspects of the software systems.

INT1 responded saying "Not all Non-functional properties can be easy to interpret. Execution time and Performance can be easily interpreted. In SBST the representation of the problem is very important and this again depends on how we define the problem..." INT2 responded by saying that "With the whole process of testing being automated there needs to be some kind of metrics that guide the search, therefore it has to be something which can be measurable automatically and quickly. Performance is easy to measure, measuring energy can be problematic but can be done through simulation..." INT3 had the following response, "I would count Security too, although Security is considered as an independent whole field. Execution Time and Energy are probably driven by the practical needs. However, I don’t see any problem as an engineer. 

Looking at the responses given by the interviewees it is understood that two factors have governed to this extensive research on those Non-functional aspects of the system. They are:

1. Ease of Interpreting the problem
2. Practical needs of the customer (end-user).

It has been discussed in section 2.1 that representing the problem is one of the core ingredients for the application of Search-based testing. All interviewees in SBST domain think that Performance and Execution time are comparatively easy to interpret, represent and then form the fitness functions that could guide the search. Adding to that practical needs of customer like the portable devices
Chapter 5. Results and Analysis of Semi-structured Interviews

and modern lifestyles are motivating the researchers to concentrate more on these aspects of the system.

5.2.3 Non-functional properties - Interviewees Perspective

When it comes to finding a set of Non-functional properties to be targeted for a system a lot of factors come into play. This has been discussed briefly in the previous section. The properties to be optimized change according to the requirements and also with the system under test. Keeping these factors aside interviewees from the Search-based testing domain more or less have the same set of Non-functional properties that are very important to be addressed irrespective of the system being inspected. However, due to different perspectives, some of them changed. The data for this theme has been coded from all the nine interviewees. It has been categorized into two parts with people working with Search-based testing techniques and ones using Regular Non-functional testing approaches. INT1, INT2 and INT3 having technical expertise in SBST say that Performance is the most important attribute of any system. Other attributes of performance like robustness and efficiency are combined and targeted for a system under test. Apart from Performance, Efficiency and Robustness they opt for Reliability and Safety properties of the system. But the third interviewee involves testability and interoperability of the system. He thinks that being testable is very important and targeting it in the first place is highly advantageous.

Looking at the data collected from the practitioners, Performance and attributes like Robustness, Efficiency, scalability are of prime concern for all the interviewees. Four out of six industrial practitioners have usability on the list. The reason for this being the web-applications as their system under test. They think that the concern for look and feel of the web applications is growing and it is very important for them to deal with this issue. Apart from this, the author found out some interesting points with respect to these properties. While interviewing the participants they were provided with a tabular information of Non-functional properties and their respective meaning (refer to Table 2.1). Most of the participants highlighted a few relations between these properties. INT1 in this regards says "There is an overlap between the properties of the system", INT5 "Looking with a technical perspective Safety and Security belong to the same category actually.". Five out of nine interviewees think that Safety and Security both can be merged with each other and targeted as one entity. Two interviewees think that Safety/Security traditionally belong to functional testing in the sense that it is something that needs to be developed/built in the system. The data collected from all the interviewees has been tabulated below.

Summary: Overall, different factors govern the selection of Non-functional properties for a particular system. In this subject of NFSBST, performance is very popularly optimized or tested and then other properties like usability, testability,
etc are gaining importance.

Table 5.1: Participants response on importance of the Non-functional properties

<table>
<thead>
<tr>
<th>Interviewee ID</th>
<th>Non-functional properties</th>
</tr>
</thead>
<tbody>
<tr>
<td>INT1</td>
<td>Performance, Reliability, Efficiency, Robustness, Safety and Availability</td>
</tr>
<tr>
<td>INT2</td>
<td>Performance, Reliability, Efficiency, Robustness, Security and Safety</td>
</tr>
<tr>
<td>INT3</td>
<td>Performance, Efficiency, Testability, Reliability, Interoperability</td>
</tr>
<tr>
<td>INT4</td>
<td>Performance, Testability, Efficiency, Maintainability, Re-usability (in terms of source code usage)</td>
</tr>
<tr>
<td>INT5</td>
<td>Usability, Performance, Robustness, Scalability, Efficiency</td>
</tr>
<tr>
<td>INT6</td>
<td>Performance, Robustness, Reliability, Usability, Scalability</td>
</tr>
<tr>
<td>INT7</td>
<td>Usability, Portability (cross-browser compatibility), Performance, Security, Safety</td>
</tr>
<tr>
<td>INT8</td>
<td>Robustness, Performance, Security, Scalability, Availability, Usability</td>
</tr>
<tr>
<td>INT9</td>
<td>Security, Performance, Usability, Portability, Maintainability</td>
</tr>
</tbody>
</table>

5.3  RQ3 - Challenges encountered employing NFS-BST

In this section, results to the third research question RQ3 using the data collected from the interviews is presented. In the most generic terms challenges are the issues or problems faced. So these words are used interchangeably. In this study, the challenges related to the application of Search-based testing techniques to Non-functional system properties are gathered and analyzed. On the whole 10 challenges have been found related to the fitness function, the metrics involved and the development phenomenon of the properties.

This section is further sub-divided into two sections where the first section concentrates on elucidating the challenges found out and then in the next section results regarding the approaches/ suggestions given by the interviewees will be presented.
5.3.1 Challenges identified from interviews

Similar to the technique applied for the two research questions above, thematic analysis helps in identifying the challenges. The respective thematic map is represented in figure 5.5. The challenges gathered using the interviews are classified into four themes as shown in figure 5.5. The themes are Fitness function, Metrics, Development and Non-functional aspects. The theme fitness function contains data coded from three interviewees, metrics contains data coded from two interviewees, development contains data coded from one interviewee and Non-functional aspects contains data coded from three interviewees. Development and Non-functional aspects themes are merged together to form another theme called as Implementation problems. This can be understood well looking at figure 5.5.

![Figure 5.5: Challenges encountered employing NFSBST](image)

In Search-based testing there is a connection between each of the themes identified here. For better understanding let's walk through an example. There is a system under test (Mobile Application), the first step is to decide on what non-functional properties (for example, Energy consumption and Usability) to optimize. The next step is, how do I measure these properties? Here, we need to figure out the metrics. The metrics are sometimes a difficult choice because it is not always possible to find ideal set of metrics for the properties being optimized. Proxy metrics are alternative for direct metrics. Once you decide on a list of metrics, to be able to design a fitness function it is necessary to find out a way to combine all these metrics to find out which one of them is computed faster or efficiently. It is always possible to find any one of these stages to be difficult for a set of properties and hence the challenges. For the mobile applications, I might get stuck on how do I capture the energy being consumed outside the mobile. Until I find a way to measure them, it is very difficult for me to optimize them.
Such difficulties exist while trying to optimize them. This example is only used to clarify the things.

5.3.1.1 Fitness Function

In this section, the challenges related to fitness function have been listed down. All the three interviewees addressed one common challenge regarding fitness function in search-based software testing. It is about how hard it often becomes to derive an efficient fitness function that is optimum for the search. In Search-based testing, everything is inter-related to each other. Right from the finding the right metrics to the problem until forming a fitness function the latter heavily depends on the earlier phase. In this regard, the interviewees stated the following.

INT1 said that "Encoding the problem in terms of fitness function in a way for the search algorithm to work efficiently is always an issue when it comes to non-functional properties."

INT2 stated that "Dealing with Non-functional system properties can get very fuzzy. Hence it becomes very difficult to define the right fitness function. For example Performance in embedded systems has many other aspects related in itself."

INT3 said something very similar to the first two interviewees. Put in his own words, "It is difficult to find the right parameters that define the fitness function to guide the search of the problem."

5.3.1.2 Metrics

Two out of the three interviewees showed concern regarding the problems that could occur with the metrics required for the target problem. The challenges identified are listed below.

INT2 stated that "As we discussed earlier the difficulties in interpreting the non-functional properties, finding the right metrics to base the search is a challenging task. Distinguishing the properties is very much important in order to find the right metrics. Performance and scalability can seem to be the same but clearly defining them is necessary. For example, I am working on a system where I enter data and I expect it to be fast and that’s related to performance. But here Scalability can be that it still has to be fast even for large amounts of data."

According to INT3 "We often lack the tools to make accurate measurements of non-functional properties. This could often lead to the formulation of inefficient fitness functions. [...]Instruction count, the one I had used in the past, requires specific virtual environment, and adds to the actual experimentation time (i.e. the cost of obtaining instruction count is increased physical wall clock time for the experiment itself) [...]Energy is even more problematic, because you cannot separate the wattage used by a single process in any modern day machine."

Looking at what the interviewees have said regarding this issue, both of
them remain to be potential challenges.

5.3.1.3 Implementation problems

All the three interviewees highlighted challenges in the context of Non-functional aspects and development. Non-functional aspects theme here represents some generic issue related to the properties and development resembles the issue that arises during the testing process or the generating the scripts to run the tests. The respective transcripts have been presented below.

INT1 thinks that "Non-functional properties do not have generic solutions. It highly depends on the SUT and new solutions should be sought out every time. We cannot reuse the solution if the system under test is different."

INT2 reports that "For properties like usability, dealing with its automation is difficult. Or if not property in itself but if a human is involved in the loop. Is it you involve the human sometimes and then carry out the search in between? This can get very tricky."

INT2 shows concern towards another challenge to which he thinks is very generic. He said the following "You’ve got to convince somebody. If you are testing non-functional properties, you have to convince either yourself or the stakeholders or the customer that something is going to be Reliable, something is going to be usable. I think people being used to automated testing, automated usability or reliability testing has a hard driven argument. Are people understanding the outcomes of the tools and techniques used for this purpose? Is it realistic? It applies more so for Non-functional properties."

INT3 stated that "Often some properties like wall clock execution time are very brittle to measure. Even the measurement or observation of these properties can affect other aspects if the system."

An approach that INT3 thinks could help in overcoming the issue specified is "Lower level infrastructure (such as OS or even hardware) should provide a more robust way to monitor them. That would make it much easier to test and optimize them."

The tabulated list of challenges identified from the interviews can be seen in Appendix B.

5.4 Validating the challenges from literature

The whole point of conducting this research is to make a framework that is useful to the practitioners of Non-functional search-based software testing. Collecting different challenges does not solve the purpose without approaches that could potentially solve them. This motivated the author to involve the challenges identified in the interviews and gather different suggestions in solving those challenges in parallel with finding new ones (subsection 5.3.1). The thematic map for
the approaches has two sub-themes namely the NFSBST Approaches and Regular Non-functional Approaches. The author felt that it would be interesting to list out the suggestions given by the Regular Non-functional testing practitioners. A question might strike about how people with no background in Non-functional search-based testing can understand challenges identified relevant to that subject. The interviewer (author) made sure that the challenges are discussed with a generic perspective. For example, Challenge 1 (as in section 4.4) is specific to a particular system and the author applies a framework GISMOE (described in section 4.4) to find out solutions about the conflicting properties. But this could remain a challenge irrespective of the technique being used to optimize the properties. Hence, the same approach has been followed for all the challenges identified to find out relevant approaches/suggestions. However, a clear distinction is made between the suggestions given by the interviewees. Those which are purely related to Search-based testing and those which are more generic.

Each of these sub-themes has been further divided into sub-themes as per the challenge. The results of the suggestions found out from the relevant interviewees will be presented under the respective challenge (example: Challenge 1, 2, so on).

5.4.1 Approaches suggested by interviewees from SBST domain

This section presents the results of the approaches/suggestions collected from interviewees using SBST techniques to non-functional properties. All the approaches suggested here are again merged into the framework.

**Challenge 1: Conflicting Properties (GISMOE)** This challenge is about the how other non-functional aspects of the system are affected during the optimization of other properties. All the three interviewees suggested approaches for this challenge.

INT1 suggested the following "Solution lies out in experimenting with a Multi-objective fitness function."

INT2 suggested the following "Multi-objective search is one approach with which this could be solved."

INT3 suggested that "Multi-Objective Evolutionary Algorithm has been applied to SE tasks for about a decade now, and the results seems sufficiently mature".

Looking at the responses it can be observed that all three of them suggest Multi-objective algorithms to be a good approach to solve this issue.

**Challenge 2: Fitness function measurement (GISMOE)** This challenge highlights the problems of not being able to find the right metrics to form the fitness function. The fitness function is very important to have a good search.
Two interviewees suggested approaches to solve this challenge. INT1 suggested the following, "We need to have certain metrics that would be used to form the fitness function and the fitness function can be very domain or system specific. I suggest in getting to know the system well to find out the solution."

INT2 suggest the following, "Simulation is recently gaining popularity with being a way to solve this. Cloud-based platforms could also be used."

These are two different approaches that could be used to solve the issue of fitness function generation/measurement.

**Challenge 3: Test Data Generation (GISMEO)** The problem of not being able to change the perspectives constantly for the functional and non-functional aspects of the system is becoming problematic and thus becoming difficult to form test data. Two interviewees suggested approaches to solve this challenge. INT1 says that "There needs to be few properties defined that are most important for the SUT and then target them."

INT3 stated that "I think this is important, but it would be feasible only if we have a reliable measurement in our hands. I know this is not a suggestion of an approach, but a problem I see is how to reconcile scenarios that pushes non-functional requirements to extreme (i.e. a usecase that really uses up battery) and scenarios that make sense to humans (i.e. normal usage)."

These indeed are two different perspectives to approach this issue. One of them suggests to find out a set of important properties for the system and target them irrespective of other issues. While the other thinks that the way we set up the use cases for the system pretty much influence the test case generation.

**Challenge 4: Balancing exploration and exploitation** The main essence of this challenges traces back to the explanation give on the connection between themes identified for challenges (refer to subsection 5.3.1). Exploration and exploitation are two keywords used in SBST. Exploration tells how the algorithm used for search behaves in the sense that, is it measuring what it has to measure? Exploitation is being able to find the right fitness function and in turn be able to find a global optimum. Balancing these phenomenon is a challenge. Only one interviewee suggested an approach to this challenge. He says that "This is a challenge overall. Not specifically to non-functional but even to the functional testing where people do not really identify it. The challenge could be finding the appropriate search method to solve the problem."

Looking at the interviewee’s response it is not really a suggestion but it could be interpreted that trying out different search methods helps in achieving an optimum result for the problem.

**Challenge 5: Sensitivity to Environment** Some non-functional properties are so sensitive to the testing environment that even small changes to the en-
environment (programming language, target platform, compiler, etc.) can lead to very strange behaviors. Two interviewees have explanations regarding this issue. INT1 says that "We need to find some techniques which are not invasive wherein the code is not altered in order to perform testing". On the contrary, INT2 thinks that it is all about measuring the property and they being very sensitive to the environment used to test them. Source code changes do happen and that part is not really the challenge. Hence, it is inferred that having a controlled environment to run the search it optimal.

**Challenge 6: Maintaining semantic equivalence** Semantic equivalence is the phenomenon of being able to assess two different programs (fitness functions) if they show the same behavior for a set of input. For some programs maintaining this phenomenon is difficult and the correctness of the results has to be evaluated. Two interviewees data has been coded regarding this challenge. According to INT2, GISMOE is about the modification of the code but not really the testing of the software. When it comes to Genetic Improvement the search is used to identify different ways in which the program source code can be improvised rather than finding optimal test cases which is done in software testing. It can be that once the code is improvised to fulfill the requirements, testing could be done using any approach. He refers to GISMOE framework which has been developed to target this problem.

INT3 has the same explanation to this challenge. According to him, "I think that we are seeing increasing amount of evidence supporting the fact that existing test cases are a sufficiently good way to ensure semantic equivalence. GISMOE showed it...program slicing techniques such as ORBS successfully captured control/data dependencies in a program using only the given test cases."

It can be inferred that GISMOE framework could be used as a solution to counter attack the issues of maintaining semantic equivalence. ORBS (observation-Based Slicing) is a language-independent slicing technique that slices multi-language developed systems. This technique can be used to retain the equivalence irrespective of the language being used.

**Challenge 7: Connecting Energy consumption and program source code** Finding a way to connect the energy consumed by a program with the source code is one of the largest hurdles. Deep understanding on the environment is necessary to be able to measure the energy consumed. Only INT2 has a relevant solution to this issue. According to him, Simulation could help in minimizing the problems of connecting the source code with the energy the software is consuming.

**Challenge 10: Efficient fitness function** The context this issue has been mentioned in the corresponding article is during the creation of fitness function. Being able to find the right metrics and finding the right combinations among
them is always a difficult task. Two interviewee's data has been used to code this theme.
Both INT1 and INT2 think that this challenge is similar to Challenge 2 discussed above and the same solutions can be used to solve.

Challenge 11: Faulty test cases Two interviewee’s data has been used to identify relevant codes for this theme.
Both INT1 and INT2 think that this challenge is similar to Challenge 6. Hence, the same approaches apply.

5.4.2 Approaches suggested by Practitioners

Along with collecting approaches from the researchers in search-based software testing, the author discussed the challenges identified from the literature review in a generic perspective to gather suggestions from the practitioners. This could be an interesting thing to discover and find if some relation could be found between what the researchers think and what the practitioners might possibly implement.

Challenge 1: Conflicting Properties (GISMOE) Data from 2 interviewees has been coded for this particular theme.
According to INT6 "A way to solve this problem can be with the usage of partition based virtual machine which resolves the problem of power and energy consumption. For example, virtual machines like Docker."
INT9 thinks that "We can overcome this kind of challenges by following base practices. To have a proper impact analysis before making any change. Impact analysis is basically sought for functional but to avoid this challenge it has to be done for the non-functional aspects. Regression testing could also potentially solve this."

Those approaches could potentially solve the issue of conflicting properties in general.

Challenge 3: Test Data Generation (GISMOE) Data from 2 interviewees has been coded for this theme.
INT 5 thinks that "It is not really a challenge to create test cases. It is a part of the process and has to be done."
INT8 stated that "Follow the usecases that are defined and create the test cases."

It can be seen that they interpret the issue to be way more general but rather in the Search-based domain this can prove to be challenging as many factors influence the creation of test cases.

Challenge 5: Sensitivity to Environment Four interviewee data has been coded for this theme.
Chapter 5. Results and Analysis of Semi-structured Interviews

Summarizing the excerpts of the all four interviewees, they all had the same idea of performing Non-functional testing on the most stable release of the product.

Overall, a few differences exist from what has been found in the literature review and what has been collected and analyzed from the interviews. For most of the research being conducted on Performance, Execution time and energy consumption aspects of the systems, it has been found out that the selection of these attributes has been driven because of factors like the ease of interpretation, their importance correlating to the practical needs of the humans.

Few challenges identified from the literature review have the usage of Genetic Improvement as the technique to formulate the fitness function and guide the search. From the interviews, it has been found out that this technique does not involve the testing aspect of the properties but involves the modification of the source code corresponding to those properties. For a better understanding, testers can employ Genetic Improvement techniques to formulate the search and find out ways to improvise the code and then testing/measuring these properties can be performed using any testing approach. However, the author still feels that these challenges are related to the optimizing the non-functional properties and decided to include these challenges in the framework. Also, to all the challenges obtained from the literature review and the interviews the approaches to solve them have been collected from both researchers and practitioners. If a challenge already had an approach identified from the literature they have been discussed with the interviewees and other suggestions from them have been gathered too. The tabulated list of approaches gathered for each of these challenges can be found in Appendix C.

Also, as discussed earlier the challenges identified from the interviews are divided into three sub-themes. The author noticed that the challenges related to fitness function overlap with some of the challenges identified from the literature review. It can be said that it is the same problem but restricted to a particular system under test and a set of non-functional properties. One of the challenges mentioned under the Metrics theme resembles one of the challenges identified from the literature review. Similarly, some of them identified under the Implementation problems (regarding Non-functional aspects, conflicting properties) category overlap with those identified from the literature review. Overall, there is not a great extent of overlap between the challenges and new interesting ones have been found out from the literature review. Thus the information gathered from the interviews has been used to validate the framework developed thus far and further enhancements have been made. The framework that has been presented previously in the earlier section has been developed taking all these criteria into consideration.
Chapter 6
Framework

In this section the final results identified as a outcome of this thesis have been presented. This has been done to improve the understandability and readability of the document. Different sections described in this chapter presents results of the dimensions of the framework generated.

6.1 System under test

A consolidated list of System under test to which Search-based testing techniques have been applied to optimize Non-functional properties are listed below.

Table 6.1: Software Systems Under Test

<table>
<thead>
<tr>
<th>S.No</th>
<th>System Under Test</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Mobile Applications</td>
</tr>
<tr>
<td>2</td>
<td>Individual Algorithms of the software system</td>
</tr>
<tr>
<td>3</td>
<td>Randomised Algorithms</td>
</tr>
<tr>
<td>4</td>
<td>Micro-Controllers</td>
</tr>
<tr>
<td>5</td>
<td>MiniSAT Boolean Satisfiability solver</td>
</tr>
<tr>
<td>6</td>
<td>Software systems developed using C-language</td>
</tr>
<tr>
<td>7</td>
<td>Software systems developed using JAVA (Object Oriented Programming)</td>
</tr>
<tr>
<td>8</td>
<td>Software systems developed using C++</td>
</tr>
<tr>
<td>9</td>
<td>Cloud systems</td>
</tr>
<tr>
<td>10</td>
<td>Safety Critical systems</td>
</tr>
<tr>
<td>11</td>
<td>Service Oriented systems</td>
</tr>
<tr>
<td>12</td>
<td>Component based systems in Automotive domain</td>
</tr>
<tr>
<td>13</td>
<td>Embedded Systems</td>
</tr>
<tr>
<td>14</td>
<td>Train Control Management Systems</td>
</tr>
<tr>
<td>15</td>
<td>Linux based systems</td>
</tr>
</tbody>
</table>
6.2 Non-functional system properties

List of Non-functional properties to which Search-based testing techniques have been applied are listed below. The results from the literature review cover most of the properties identified. Apart from that, Testability and Wall-clock execution times are two new ones identified. Refer to Table 4.3 for further information.

6.3 Challenges identified along with corresponding approaches/suggestions

This sections presents the challenges identified both from the literature review and the interviews. Alongside, the approaches/suggestions collected from the existing literature and the researchers have been correlated. Information has been tabulated and can be seen below. For simplified reading, the challenges will be referred to with the shortcuts given to each of these challenges. For further information on the shortcuts to these challenges, refer to section 4.4.

Table 6.2: Challenges and Approaches while employing NFSBST

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Approaches/Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenge1: Conflicting Properties</td>
<td>Using Multi-objective optimization algorithms</td>
</tr>
<tr>
<td>Challenge2: Fitness function measurement</td>
<td>Finding the right metrics to form the fitness function. Deep inspection of the SUT is beneficial.</td>
</tr>
<tr>
<td></td>
<td>Simulation of the properties is gaining importance. Even cloud-based platforms could be used.</td>
</tr>
<tr>
<td>Challenge3: Test Data Generation</td>
<td>Using GISMOE framework can solve this issue.</td>
</tr>
<tr>
<td></td>
<td>Deciding on a set of important properties for the SUT and then targeting them is a good approach.</td>
</tr>
<tr>
<td>Challenge4: Balancing exploration and exploitation</td>
<td>Using Hybrid approaches can solve this issue.</td>
</tr>
<tr>
<td>Challenge5: Sensitivity to Environment</td>
<td>Finding techniques that are not invasive wherein the code is not changed frequently.</td>
</tr>
<tr>
<td>Challenge6: Maintaining Semantic Equivalence</td>
<td>Using Coevolution algorithms can be beneficial to gain confidence in the results</td>
</tr>
<tr>
<td></td>
<td>ORBS slicing techniques can be used to control data dependencies</td>
</tr>
</tbody>
</table>
### Chapter 6. Framework

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Description</th>
<th>Solution</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenge 7: Connecting Energy consumption and program source code</td>
<td>Simulation is a potential solution</td>
<td></td>
</tr>
<tr>
<td>Challenge 8: GIGI4 (Factors affecting energy consumption)</td>
<td>Genetic Programming can be used to approach this issue. Equation has to be formulated inclusive of the parameter related to energy, memory, etc.</td>
<td></td>
</tr>
<tr>
<td>Challenge 9: Improvising search space</td>
<td>Stopping the program from entering the next iteration to find other solution can be a suggestion to solve this issue.</td>
<td></td>
</tr>
<tr>
<td>Challenge 10: Efficient fitness function</td>
<td>Approaches given for Challenge 2 apply here as well.</td>
<td></td>
</tr>
<tr>
<td>Challenge 11: Faulty test cases</td>
<td>Approaches given for Challenge 6 apply here as well.</td>
<td></td>
</tr>
<tr>
<td>Challenge 12: Automotive redundancy allocation impact</td>
<td>Approaches given for Challenge 1 apply here as well.</td>
<td></td>
</tr>
<tr>
<td>Challenge 13: Encoding the problem in terms of fitness function</td>
<td>Approaches given for Challenge 2 apply here as well.</td>
<td></td>
</tr>
<tr>
<td>Challenge 14: Finding right metrics for Search</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Challenge 15: Finding appropriate tools for assessing metrics</td>
<td>–</td>
<td></td>
</tr>
<tr>
<td>Challenge 16: Execution time very brittle to measure</td>
<td>Lower level infrastructure should be robust in order to measure them effectively.</td>
<td></td>
</tr>
</tbody>
</table>

Getting all the information together an overview of the framework is provided in this section. The framework can be used in two perspectives. One is the Requirements perspective and the other is the meta-heuristics perspective. Only one perspective is presented in this section. The others can be accessed following Appendix A or Appendix F.
Chapter 6. Framework

59

Figure 6.1: Overview of the Framework

This framework has been designed to give the search-based software engineering practitioners both in academia and industry a set of guidelines or checklist that can be looked into prior to the verification and validation of the non-functional attributes of the system. Due to the page-width constraints, the framework has been represented in two different parts. In figure 6.2 and figure 6.3, we can observe the latest version of the framework.

6.4 Understanding the framework

The information in these diagrams is portrayed in the following fashion. Analysis has been made on the Requirements perspective of the framework. The topics of the mind map are the different non-functional properties that have been
identified overall. Next to the properties, there are the different System Under Test. For a unique combination of SUT and non-functional properties there are a set of meta-heuristics applied to optimize or test the properties. Finally, there are the different challenges encountered during this process. For a few challenges, there are approaches/suggestions provided as well. These strategies are depicted with the acronym AS ("Approaches to Solve").

**Example:** The following is an example illustrating how the author intends the framework to be used. A search-based testing practitioner interested in optimizing power consumption of the system. He could then refer to this framework and then check for power/energy consumption section and can then search for the different systems in which this property has been optimized. Once the systems have been identified, then for each system the meta-heuristic techniques used is available. Now that the Non-functional property, the system, and the techniques are known, the practitioner can know different challenges that might show up during the execution. For these challenges, approaches/suggestions have been listed too. In this way the practitioner can get well prepared prior to the optimization of Non-functional properties.

The following table consists of the list of articles that have contributed to the generation of the framework. This could be used by the users of the framework to find a link back to the original source which helps in increasing the understanding pertaining to a claim or suggestion. This is also helps in increasing the trustworthiness of the solutions provided.

**Table 6.3: Articles used in the framework**

<table>
<thead>
<tr>
<th>Article ID</th>
<th>Article Title</th>
</tr>
</thead>
<tbody>
<tr>
<td>P1</td>
<td>The GismoE challenge: Constructing the pareto program surface using genetic programming to find better programs (keynote paper). [24]</td>
</tr>
<tr>
<td>P2</td>
<td>Applying particle swarm optimization to quality-of-service driven web service composition. [47]</td>
</tr>
<tr>
<td>P3</td>
<td>How to solve it: modern heuristics. [53]</td>
</tr>
<tr>
<td>P4</td>
<td>Evolutionary improvement of programs. [69]</td>
</tr>
<tr>
<td>P5</td>
<td>Reducing energy consumption using genetic improvement. [6]</td>
</tr>
<tr>
<td>P6</td>
<td>Gi4gi: Improving genetic improvement fitness functions. [26]</td>
</tr>
<tr>
<td>P7</td>
<td>Automated design of algorithms and genetic improvement: contrast and commonalities. [21]</td>
</tr>
<tr>
<td>P8</td>
<td>Redundancy allocation in automotive systems using multi-objective optimisation. [51]</td>
</tr>
<tr>
<td>P9</td>
<td>Searching for pareto-optimal randomised algorithms [54]</td>
</tr>
</tbody>
</table>
Figure 6.2: Non-functional Search-based Software Testing Framework (Part1)
Chapter 6. Framework

Figure 6.3: Non-functional Search-based Software Testing Framework (Part 2)
Chapter 7

Discussion

This chapter provides a comprehensive discussion on what has been found as a result of conducting this research. The analyzed results were further validated if they are in coherence with the findings of the literature review. This helps in building trust in the reader and rely on the results of the research. Discussions in this chapter will be made on each of the research questions formulated for this research.

7.1 Software systems being tested using SBST

Various software system under test to which Search-based testing techniques addressing the non-functional system properties are collected by carrying out Literature review and Semi-structured interviews. The list of software systems under test are presented in section 6.1.

From a large variety of SUTs identified in this thesis, the author has observed that SBST to non-functional system properties can be applied to any piece of software. Right from a simple function to a complete system, there are instances of applying SBST to optimize these properties. In [69] a simple C function is taken to which techniques have been applied to investigate the problem of semantic equivalence. The author’s main intention here is to find out different kinds of systems and then develop the research starting from this. It has been constantly stated in the document that the framework is the end outcome of this study and this aspect is an important component of the framework. There are no interesting patterns observed between the kinds of systems being tested using SBST to optimize non-functional properties. But one interesting thing identified was that irrespective of the software being tested SBST can be scalable. Another interesting aspect found out as a result of conducting interviews is that the system under test deeply influences the selection of the non-functional properties to be tested or optimized. However, through literature review little did the author know about this as most of the research just focus on the system under test and a selected set of properties being optimized.


Chapter 7. Discussion

7.2 Non-Functional System properties tested

A range of Non-functional properties are being targeted using the Search-based testing techniques. The list of properties that have been subjected to the optimization process using SBST are presented in section 6.2.

From the available list of Non-functional properties, the author observed that Performance, execution time and power/energy consumption properties of the systems have been the core interests of many researchers. While trying to find the actual reason behind these properties gaining a lot of importance, this is what the author has found. A lot of factors affect the selection of non-functional properties targeted. The factors are the System under test and the domain, the resources that are available to carry out the testing process and finally the stakeholders perspective. Very few studies involved in the literature review highlights these factors influencing the selection of properties. Now that we have known the factors, the reasons identified for an extensive research being carried out on the properties mentioned above are, Performance is a very important aspect of any system and it has to be addressed. It is stated in [6] that due to the growth in the computing infrastructure and the ecological issues concern towards the energy consumption of the software is gaining importance. Looking at the interviewee excerpts presented in the previous section, the reasons for this are: 1) They are comparatively important taking into consideration the growing practical needs 2) They can be easily interpreted and finding metrics for them is easy when compared to other non-functional properties. It is not argued that other properties are not so important. It is just that there is very little evidence to come to some conclusion.

Also, while discussing about the three popularly tested properties in SBST the author noticed that defining the properties is equally important. With deeper inspection regarding this, all the interviewees except for two practitioners think that the way the non-functional properties are defined influence the way in which testing is done. What is interesting to know is that all the researchers in SBST think that defining is very important as it affects other factors like finding the metrics to the properties and then finding the correct fitness function to guide the search. In an attempt to find out if the same applies to non-functional testing in general, four out of them (interviewees with experience in Non-functional testing) accept that it is important. Two of them, think it is not so necessary to define the properties and the business use cases are good way to proceed.

It is observed that most of the articles involved in the literature review stress on these properties in general and this data is backed up by the data collected from the interviews where they accept for that patterns observed and even the practitioners when asked about some important set of properties that are to be addressed, Performance and other attributes like Reliability, Efficiency are mentioned in the list. It can be said that the three properties are given more im-
importance comparatively not that the other are given less importance but factors discussed above come into play.

7.3 Challenges faced while employing NFSBST

The challenges that affect the process of testing non-functional properties using SBST have been gathered using Literature review and Semi-structured interviews. Challenges identified from the different data sources are presented in section 6.3.

Some challenges identified from the Literature review and the interviews were related to the fitness function. Articles [21, 24] support the argument by portraying respective challenges. Even the interviewees’ initial response on the challenges encountered was related to fitness function. The non-functional properties being very hard to interpret, give a hard time deriving an efficient fitness function. To overcome issues as such, it is important to find out right metrics and this could be a whole new field in itself. Evidences from the literature and excerpts from the researchers suggest to deeply inspect the system and then have a clear idea on what has to be measured. Regular checkups should be made if the properties being measured are indeed the ones that are to be measured.

Another set of challenges identified were related to finding the right metrics for the search algorithm. Some of the challenges identified are about improvising the search space, maintaining semantic equivalence, finding right metrics to base the search, misinterpretation of non-functional properties leading to the formulation of inefficient fitness functions, etc. Articles [21, 69] provide evidences of these being a challenge employing NFSBST. Discussion around the extent of overlap between the challenges would not be interesting as each of them pose a problem in the respective domain. However, for a few challenges like the misinterpreting the non-functional properties leading to inefficient fitness function can be much of a generic SBST challenge.

What has not been highlighted much in the literature review are challenges that are more generic to the application of NFSBST. Some of the challenges related to usability (its automation) have been discussed by the researchers but have not really been addressed in most of the articles involved in this literature review. But [2] addresses usability criteria of the software systems in terms of the meta-heuristics being used to measure it and the challenges faced while optimizing it. In the context of application of meta-heuristics it is related to the construction of covering arrays. Clear description can be seen in that systematic literature review.

After the collection of the Literature review results the author found out a few issues that use Genetic improvement as a meta-heuristic. Genetic Improvement, a variant of Genetic Programming deals with the improvisation of the code using the search but not really with the testing aspect of the system.
This information has been obtained on having a detailed discussion with one of the interviewees. Other interviewees still think that they somehow belong to the testing process as source code changes are involved during the optimization of a system in terms of either its non-functional or functional properties.

To all those challenges identified, collecting a few approaches/suggestions from the researchers would make the framework complete. Different perspectives have been collected to all the challenges identified as a part of this research. Along with the researchers in the field of SBST, practitioners also contributed some of their ideas to solving some of these problems. A solution to solve a problem does not always rely on the core components involved in the subject. Sometimes, even the simplest of things might solve the problem and a few suggestions given by these practitioners might help and further improve the framework. Finding out the strategies to solve them has been detailed in Chapter 6. The approaches have not been identified to all the challenges addressed in this research. There are several reasons to it. They are not like puzzles to which you find out a piece and solve it. They are rather open questions which require discussions and there are no easy answers. The author has not been able to find out the meaningful approaches backed up by evidence. Some reasons are:

1. There are no hard and fast rules to solve these challenges
2. Due to the lack of evidence from the literature sources
3. None of the interviewees brought up this discussion. Reasons for this might be:
   (a) They think some are trivial
   (b) They have never come across such challenges
   (c) Some challenges are completely pertained to the study from where it has been identified from

Summarizing on what has been discussed above, the author was able to find out different software system under test to which Search-based testing techniques are being employed to test and optimize the non-functional system properties. Next, an inspection on non-functional properties targeted using these techniques has been made. A connection between the systems and the non-functional properties identified has been maintained. The main intention is to create a connection between them and be able to reflect that in the framework (chapter 6). For a particular system and a set of non-functional properties, potential challenges that could affect the optimization/testing process have been identified. Then approaches to solve some of these challenges have been gathered. Tracing back to the results presented in different sections by carrying out different research methods it can be understood how the framework has evolved in two phases and validated. The version that is presented in chapter 6 is the latest one after making all the respective changes.
7.4 Implications for research and practice

This study presents a framework with information that has been gathered from the existing literature and researchers working in this field. A set of SUT and the respective non-functional properties of interest along with challenges have been presented. Approaches/suggestions that could potentially solve the challenges identified are also a part of the framework. Trustworthiness of the approaches/suggestions is one important factor that needs to be investigated. These approaches have been identified both from literature sources and the interviewees. To the approaches identified from the literature, a link back to the original source has been mentioned both in the document and the framework. This improves the confidence of the reader in relying on a particular suggestion, as it has been implemented by someone and there is some evidence. But when it comes to the ones identified from the interviewees, they are a bunch of opinions from different individuals. These suggestions exist and the target audience are encouraged to try them out. They are potential ways of solving the challenges but not any hard and fast rule. This framework could be used as starter guide for practitioners interested in employing Search-based testing techniques to Non-functional system properties. The results can be used by them to avoid some of the barriers that may arise. The target population for this research are the industrial practitioners employing NFSBST. Any novice practitioner can use the findings of this study and get prepared prior than stumbling upon some issues and breaking heads to find an potential solution to the problem. Considering the academic perspective, this study has been inspired from the work of Afzal et al. [2]. This research has taken into account literature that have been published after this SLR. This consolidate information might even help the researcher to know the extent to which NFSBST has been applied. The author also tried to align this framework with some quality standards. For this research, ISO 25010 [31] has been used to define the Non-functional involved in this research. A discussion with the interviewees has been made while discussing about the framework to understand their interpretation. Further alignment can be done which is discussed in the future work.

7.5 Threats to Validity

Trustworthiness of the findings in qualitative software engineering research is always a topic of discussion [63]. The chosen research methods - Literature review and Semi-structured interviews are one of the empirical qualitative methods. Identifying the validity threats and employing certain mitigation strategies helps in ensuring the quality of the research done. For qualitative research methods, like the one applied for this study four types of validity threats are to be addressed [13]. Validity threats that were relevant for this study have been dis-
cussed in detail along with the steps taken to avoid the threats and ensure good quality in the research.

7.5.1 Construct Validity

Construct validity deals with the differences that exist between what the researcher intends to achieve and what has been actually investigated in relation to the research questions [61]. One possible chance of facing this validity threat is during the collection of information using the interviews. Misinterpretation of the interview questions by the interviewees leads to gathering irrelevant information. This has been avoided by conducting a pilot interview to assess the understandability and relevance of the questions. Furthermore, the interview instrument has been revised with the supervisors feedback to ensure the validity. The feedback has been used to make tiny tweaks to the instrument. To get accurate data the interviewer developed a support guide (manuscripts) for the interviewees. Hence, both the participants being on the same page, accurate information could be collected.

While performing qualitative research, it is necessary to triangulate the data [61]. Keeping this in mind while performing the researchers, for instance, interviews from the practitioners has also been considered to triangulate the data. However, a limitation of this study is that only single data collection method has been used. Another limitation of this study is finding relatively less number of interviewee in the Search-based testing domain.

7.5.2 Internal Validity

Internal validity threats refer to the finding of factors that might affect the outcome of the study [61]. Sometimes the changes go unnoticed due to the fact that these changes occur due to some third level factors.

Possible chances of facing these threats are higher if the researcher is not conscious of the treatment being applied to the research. Improper selection of the literature, interview respondent selection and misinterpretation of data. To mitigate the risk of having internal validity, the author has employed a structured approach to the selection of data for the literature review.

The interview respondents selection also affects the data being collected. Hence, the subjects with relevant experience have only been interviewed. Convenience sampling has been used to select the subjects and all the subjects have more than 2 years of experience. In the case of researchers interviewed for this research, their expertise in this domain has been more than 8 years. This reduces the chances of misinterpretation in the questions.

Poor data analysis leading to incorrect solution is yet another risk for this research. To complement this risk countermeasures have been taken. After the detailed analysis is done a discussion with the supervisor is made to confirm the
validity of the findings. There also exists another possible threat to internal validity due to conducting a literature review for the comparative analysis made in chapter 5 rather than a systematic literature review [37]. Chance of missing out some relevant literature has not been considered for this study.

7.5.3 External Validity

External validity refers to the extent to which the results obtained be generalized. This threat has been mitigated by selecting the most relevant sample. The sampling used for this research is convenience sampling. It is not argued that this is the representative sample but considering constraints like the resources and time available the sample selected is relevant to the field of research. Also, the results are valid since the interviewees have over 8 years of experience in this domain. The results can be generalized to research to both researchers in Search-based software testing and also to practitioners implementing Non-functional Search-based software testing. The other set of interviews conducted include a heterogeneous sample with respect to their experience in Non-functional testing and their role in the organizations.

7.5.4 Reliability

During the study, to ensure the reliability of the findings, the methods being used have been constantly pretested. Pretests were a part of this study to keep a check on the collection methods not biasing the end results. The interview instrument has been piloted with one interview and suggestions from the supervisor and the participants were constantly used to keep a check on the reliability of the findings.
Chapter 8

Conclusions and Future Work

8.1 Conclusions

This study focused on identifying different dimensions of Non-functional Search-based software testing. They include the systems to which the Search-based testing techniques have been applied, the prime non-functional properties of interest, several challenges that could hinder the process of optimizing these properties. To the challenges identified a list of approaches/suggestions found from the literature and other researchers have been gathered. In this study, along with a literature review, nine semi-structured interviews were conducted to identify all the dimensions (see chapter 6) discussed above. It must be noted that this subject has mostly been restricted to research than being implemented in the industry.

In relation to RQ1 - software systems under test, the most interesting findings are: Not only NFSBST but SBST in general can be applied to software systems of any scale. It has been applied to different software systems ranging from cloud-based systems to a simple function written in C language. Around 14 different types of software systems have been identified in this study.

The findings regarding RQ2 - Non-functional properties being tested in this domain, these are the most prominent results. Several factors govern the selection of non-functional properties being targeted for a specific System under test. The factors are: 1) The SUT itself. 2) The stakeholders perspective. 3) The resources available. Apart from this, defining the Non-functional (interpretation) also has a major role in the testing process. This is an important issue to those employing SBST while it is not so much important to those working on non-functional testing. The reason for this is that the way the problem is represented is one of the key factors in search-based testing. Depending on the representation the metrics would be derived and eventually the fitness function will be formulated. If the interpretation goes wrong, the measurement will not be accurate.

When it comes to non-functional testing in general, Performance, Execution time and Energy consumption are three most popularly tested non-functional properties in this domain. The reasons for this amount of extensive research are
that these attributes are relatively easy to interpret. Apart from that taking into account the daily practical needs of the humans, these are the properties that pretty much rule the software being used. To triangulate the data even the practitioners involved in this research find performance of the systems to be one of the important attributes of the system.

Discussing about RQ3 - Challenges faced employing NFSBST, three main phenomenon of Search-based software testing are the source of many problems. Basing on the discussion made in subsection 5.3.1, the three main phenomenon are: 1) Fitness function 2) Metrics 3) Implementation problems. The challenges identified have been a combination of all these. Most important point to remember is that sometimes the challenge remains to be the same but in SBST it is very specific to the system under test. Apart from that, interesting ones related to other properties like Usability, Scalability have been brought to the notice by the interviewees. They think that these properties are slowly gaining importance and testing (automation) can be really hard sometimes. Other part of this research questions is about finding out potential solutions to the challenges identified. Approaches have been found using both the methods used in this research.

Overall, the information gathered is mapped together to form a theoretical framework that can be viewed in two different perspectives and hopefully this helps the practitioners employing NFSBST.

8.2 Contributions to research

This study adds to the existing body of knowledge in the field of Search-based software testing. This thesis builds up using the ideas of the authors of [2] and adds a new dimension to it. As opposed to prior research in this subject, this study tries to discover all possible fields that could potentially make this framework useful to the target population. The information brought together with the aid of efficient techniques and tools to diagrammatically portray the information and make it more usable.

8.3 Future Work

The future work of this study can be to identify additional people who have the relevant expertise in Non-functional Search-based software testing and interviewing them to find out additional information. Ideal case would be if the sample population included of some practitioners in this domain. Once the framework has been enhanced with sufficient information, the next step would be to have this framework used by the practitioners and find out some interesting things about the framework. Aspects like, usability of the framework and additional information/dimensions to the framework could be investigated. Thus,
the author hopes that the investigation by experimenting with this framework helps in getting a clear understanding of the concepts of Non-functional Search-based software testing. Also, the framework could be aligned with standards on Non-functional testing and Non-functional properties.
References


References


References


[73] Y. Zhang, M. Harman, and A. Mansouri. The SBSE repository: A repository and analysis of authors and research articles on search based software engineering. crestweb.cs.ucl.ac.uk/resources/sbse_repository/.
Appendices

Appendix A: Using the Framework

In this section the instructions on how to use the framework is highlighted. Use the link given in this section to browse on the Internet. You are initially redirected to the overview of the framework, mentioned in chapter 6. Then you have two ways in which you can handle this framework. You can either use this framework from the meta-heuristic used to solve the testing process or the non-functional system properties involved.

- **STEP1:** Open the overview.
- **STEP2:** Click the links provided in the topics. You can choose one of the perspectives mentioned above.
- **STEP3:** You will be redirected to the respective maps where the details have been provided.
- **STEP4:** Use the framework in the fashion based on the instructions specified in chapter 6.

The link to the framework is available here at NFSBST Framework.
## Appendix B: List of challenges

**Table B1:** List of challenges identified from semi-structured interviews

<table>
<thead>
<tr>
<th>Theme (Challenge Area)</th>
<th>Challenge identified</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fitness Function</td>
<td>1. Encoding the problem in terms of fitness function</td>
</tr>
<tr>
<td></td>
<td>2. Difficulty defining the right fitness function</td>
</tr>
<tr>
<td></td>
<td>3. Finding right parameters to formulate the fitness function is challenging</td>
</tr>
<tr>
<td>Metrics</td>
<td>1. Finding right metrics to base the search is difficult</td>
</tr>
<tr>
<td></td>
<td>2. Appropriate tools are not available that help in finding out right metrics and assess</td>
</tr>
<tr>
<td></td>
<td>them</td>
</tr>
<tr>
<td>Implementation problems</td>
<td>1. Non-functional properties do not have generic solutions</td>
</tr>
<tr>
<td></td>
<td>2. Properties like usability are hard to automate or involving human in the testing pro-</td>
</tr>
<tr>
<td></td>
<td>cess.</td>
</tr>
<tr>
<td></td>
<td>3. Convincing somebody on a decision</td>
</tr>
<tr>
<td></td>
<td>4. Properties like execution time are brittle to measure.</td>
</tr>
</tbody>
</table>
Appendix C: Approaches/Suggestions collected using Interviews

Table C1: Approaches/Suggestions collected from semi-structured interviews

<table>
<thead>
<tr>
<th>Challenge</th>
<th>Approaches/Suggestions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Challenge 1 - Conflicting Properties (GISMOE)</td>
<td>1. Multi-objective algorithms could potentially solve this issue</td>
</tr>
<tr>
<td>Challenge 2: Fitness function measurement (GISMOE)</td>
<td>1. Find out the right metrics that would be used to form the fitness function. Understand the SUT very well.</td>
</tr>
<tr>
<td></td>
<td>2. Simulation is recently gaining popularity with being a way to solve this. Cloud-based platforms could also be used.</td>
</tr>
<tr>
<td>Challenge 3: Test Data Generation (GISMOE)</td>
<td>A set of properties must be defined prior after a deep inspection on the SUT and targeting them resolves this issue.</td>
</tr>
<tr>
<td></td>
<td>Having clear-cut use cases influences the test case generation. This clears the confusion and set a concrete path to test case generation.</td>
</tr>
<tr>
<td>Challenge 4: Balancing exploration and exploitation</td>
<td>Experimenting different search methods is the possible solution.</td>
</tr>
<tr>
<td>Challenge 5: Sensitivity to Environment</td>
<td>Finding techniques which are not invasive wherein the code is not altered to perform testing.</td>
</tr>
<tr>
<td>Challenge 6: Maintaining semantic equivalence</td>
<td>1. Using GISMOE framework to solve this problem</td>
</tr>
<tr>
<td></td>
<td>2. Program slicing techniques like ORBS can also be used as a contingency plan.</td>
</tr>
<tr>
<td>Challenge 7: Connecting Energy consumption and program source</td>
<td>Simulation is possible solution to mitigate this issue.</td>
</tr>
</tbody>
</table>
Appendix D: Snowballing Results - Articles found

Here, the tabulated information about the articles examined during the snowball sampling approach is presented. The iteration results are as shown below. The abbreviations to the acronyms used are also mentioned.

- BS- Backward Snowballing
- FS- Forward Snowballing
- ET- Exclusion based on Title
- EA- Exclusion based on reading the Abstract
- EF- Exclusion based on reading the Full-text
- EL- Exclusion based on Language
- FN- Full-text unavailable
- IT- Inclusion based on reading the Title
- IA- Inclusion based on reading the Abstract
- IF- Inclusion based on reading the Full-text

**Table D1:** First Iteration Results

<table>
<thead>
<tr>
<th>ID</th>
<th>Number of references and Citations examined</th>
<th>Excluded Papers</th>
<th>Included Papers</th>
</tr>
</thead>
</table>
| P1 | BS: 16  
     FS: 0 | ET: 16 | -  
     BS: 16  
     FS: 0 | -  
     BS: 16  
     FS: 0 | -  |
| P2 | BS: 68  
     FS: 0 | ET: 55  
     EA: 10  
     EF: 3  | -  
     ET: 55  
     EA: 10  
     EF: 3  | -  
     ET: 55  
     EA: 10  
     EF: 3  | -  |
| P3 | BS: 19  
     FS: 0 | D: 1  
     ET: 11  
     EA: 6  | -  
     D: 1  
     ET: 11  
     EA: 6  | -  
     D: 1  
     ET: 11  
     EA: 6  | -  |
| P4 | BS: 22  
     FS: 0 | ET: 17  
     EA: 5  | -  
     ET: 17  
     EA: 5  | -  
     ET: 17  
     EA: 5  | -  |
| P5 | BS: 129  
     FS: 11 | D: 3  
     ET: 115  
     EA: 5  
     EF: 4  | T: 10  
     EA: 1  | IT: 1  
     IA: 1 | -  |
| P6 | BS: 39  
     FS: 7 | ET: 39  
     EA: 1  | ET: 6  
     EA: 1  | -  
     ET: 39  
     EA: 1  | -  |

References
### Table D2: Second Iteration Results

<table>
<thead>
<tr>
<th>ID</th>
<th>Number of references and Citations examined</th>
<th>Excluded Papers</th>
<th>Included Papers</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>BS</td>
<td>FS</td>
</tr>
</tbody>
</table>
| P14 | BS: 40  
FS: 67 | ET: 16 | D: 3  
EA: 3  
EL: 1  
FN: 1 | -   | -   |
|     |                                            | ET: 59 | D: 2  
ET: 160  
EL: 1  
FN: 2 | -   | -   |
| P15 | BS: 14  
FS: 165 | ET: 55 | D: 3  
ET: 59  
EA: 3  
EL: 1  
FN: 1 | -   | -   |
|     |                                            | ET: 10  
EF: 3 | D: 2  
ET: 160  
EL: 1  
FN: 2 | -   | -   |

**References**
Start Set

As a result of implementing the Snowballing Approach to find the articles to conduct the Literature Review, a start set of 13 articles were chosen from a candidate set of 62 articles. The search string and the database used for retrieving these articles have been mentioned in the previous sections. Table 8.5 is the list of articles in the start set.

<table>
<thead>
<tr>
<th>Article ID</th>
<th>Articles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Page</td>
<td>Reference</td>
</tr>
<tr>
<td>------</td>
<td>-----------</td>
</tr>
</tbody>
</table>
Iteration 1

Backward Snowballing

For the first iteration of Backward snowball sampling of the start set, a total of 550 references had to be carefully inspected and passed where the filtration of the articles has been done in the following fashion. The whole list of references has been collected into a spreadsheet after which the titles of each of these articles is read. 427 articles have been removed from the using this criterion. The inclusion and exclusion criteria have been simultaneously verified. Few entries have been assigned the label "Might be Relevant" when the author has inspected the title of the reference using the aforementioned criteria and was convinced that this article could be relevant. The abstracts of these articles were read, then 35 articles have been removed. Along with these, there were a few references that were duplicates or those which have been already dealt with. There were 13 duplicates and these have also been excluded for the iteration process. One of the articles had no full-text availability and another reference was that of an online Search-based Software Engineering repository and these have been excluded as well.

The next step in the inspection was done in a bit more detail. There were 11 articles that the author decided to apply the detailed inspection criteria on. The full-text was read for each of these articles and finally, only 4 articles have been included for the next iteration. The reason for exclusion is also explicitly maintained by the author for future reference. The four articles are tabulated in the Table 8.6 below.

Table D4: Iteration 1 - Backward Snowballing Results

<table>
<thead>
<tr>
<th>Article ID</th>
<th>Articles</th>
</tr>
</thead>
</table>
Forward Snowballing

Same approach used for backward snowball sampling has been applied for forward snowball sampling. A total of 327 references had to be inspected. The exclusion of the articles is done as follows. 275 citations have been excluded based on the titles. The abstracts have been read for all those citations whose title seemed to be relevant where 21 citations have been excluded. There were 6 duplicate citations and 16 citations have been excluded as they were published in different languages (not English). For 6 articles there was no availability of the full text and they have been excluded as well.

The author was finally left with 5 unique articles where the full text has been read and only 1 articles has been included for the next iteration. The details are given in the following Table 8.7.

Table D5: Iteration 1 - Forward Snowballing Results

<table>
<thead>
<tr>
<th>Article ID</th>
<th>Articles</th>
</tr>
</thead>
</table>

Overall 5 articles have been selected for the next iteration for which the same approach has been applied. The results for the first iteration are depicted in the tables above.

Iteration 2

Backward Snowballing

During the second iteration of Backward Snowballing 166 references had to be inspected among which 147 were removed upon examining the titles, 11 papers were duplicates which were already examined, 2 articles had no full-text readability and 6 were excluded upon reading the abstracts.

8.3.0.1 Forward Snowballing

During the second iteration of Forward Snowballing 312 citations had to be reviewed, where 273 were excluded upon inspecting the titles, 12 were duplicates, 5 articles were published in different languages, 4 articles had no full-text readability and 6 articles were excluded upon reading the abstract. The results of the second iteration are tabulated in tables above.
Appendix E: Articles found using the Search-based software engineering repository

As mentioned earlier in the literature review section the Search-based repository [73] has also been inspected to find even more unique articles. A total of 9 unique articles were found and most relevant information has been gathered from 7 of the whole set. They are tabulated below.

**Table E1: Articles found using the SBSE repository**

<table>
<thead>
<tr>
<th>Article ID</th>
<th>Articles</th>
</tr>
</thead>
</table>
Appendix F: NFSBST Framework

References: This figure depicts the references displayed used in the framework.

**Figure F1:** Overview of the Framework

The next two figures represent the meta-heuristics’ perspective of the framework.
Figure F2: Non-functional Search-based Software Testing Framework–Meta-heuristics’ perspective (Part 1)
**Figure F3:** Non-functional Search-based Software Testing Framework- Meta-heuristics’ perspective (Part 2)