INTERACTIVE SEARCH-BASED SOFTWARE TESTING
Development, Evaluation, and Deployment

Bogdan Marculescu
Interactive Search-Based Software Testing
Development, Evaluation, and Deployment

Bogdan Marculescu
Interactive Search-Based Software Testing
Development, Evaluation, and Deployment

Bogdan Marculescu

Doctoral Dissertation in Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SWEDEN
Interactive Search-Based Software Testing

Development, Evaluation, and Deployment

Bogdan Marculescu
Interactive Search-Based Software Testing
Development, Evaluation, and Deployment

Bogdan Marculescu

Doctoral Dissertation in Software Engineering

Department of Software Engineering
Blekinge Institute of Technology
SWEDEN
Contents

Abstract xi
Acknowledgements xiii
Overview of Publications xv

1 Introduction 1
  1.1 Overview .............................................. 1
  1.2 Context ............................................ 3
  1.3 Our industrial partner ............................... 5
  1.4 Related Work ....................................... 8
  1.5 The Interactive Search-Based Software Testing System .... 12
  1.6 Research Questions ................................. 14
  1.7 Research Methods .................................. 17
  1.8 Summary of Papers Included in the Thesis .......... 19
  1.9 Discussion ........................................ 26
  1.10 Threats to Validity ............................... 36
  1.11 Future Work .................................... 37
  1.12 Conclusions .................................... 39

2 A Concept for an ISBST System 41
  2.1 Introduction ...................................... 42
  2.2 Background ...................................... 42
  2.3 Search-Based Testing System ..................... 43
  2.4 Validation and Discussion ....................... 46
  2.5 Conclusions ..................................... 46
3 Objective Re-Weighting to Guide an Interactive Search Based Software Testing System 47
  3.1 Introduction ................................................. 48
  3.2 Related Work .................................................. 49
  3.3 Industrial Context ............................................ 49
  3.4 Objective and Method .................................... 51
  3.5 Implementation .............................................. 55
  3.6 Evaluation ..................................................... 56
  3.7 Conclusions ................................................... 59

4 Practitioner-Oriented Visualization in an Interactive Search-Based Software Test Creation Tool 61
  4.1 Introduction ..................................................... 62
  4.2 Related Work .................................................. 63
  4.3 Industrial Context ............................................ 64
  4.4 The ISBST System ........................................... 66
  4.5 Empirical Evaluation ....................................... 68
  4.6 Discussion .................................................... 71
  4.7 Conclusions ................................................... 72

5 An Initial Industrial Evaluation of Interactive Search-Based Testing for Embedded Software 73
  5.1 Introduction ..................................................... 73
  5.2 Related Work .................................................. 75
  5.3 Case Study Overview ...................................... 78
  5.4 The ISBST System ........................................... 81
  5.5 Industrial Evaluation ....................................... 89
  5.6 Laboratory Experiment ................................... 94
  5.7 Discussion .................................................... 98
  5.8 Validity Threats ............................................. 100
  5.9 Conclusions and Future Work .............................. 101
  5.10 Acknowledgements ........................................... 102

6 Tester Interactivity makes a Difference in Search-Based Software Testing: A Controlled Experiment 105
  6.1 Introduction ..................................................... 105
  6.2 Context .......................................................... 107
  6.3 Experimental Design ...................................... 116
  6.4 Results .......................................................... 127
# CONTENTS

6.5 Discussion .................................................. 140
6.6 Threats to Validity ........................................ 143
6.7 Conclusions ................................................. 145
6.8 Appendix .................................................... 146

7 Using Exploration Focused Techniques to Augment Search-Based Software Testing: An Experimental Evaluation 149
  7.1 Introduction ............................................... 149
  7.2 Research Questions ...................................... 152
  7.3 Experimental Setup ...................................... 153
  7.4 Results and Analysis .................................... 163
  7.5 Discussion ............................................... 170
  7.6 Conclusions .............................................. 172
  7.7 Future Work .............................................. 172

8 Transferring Interactive Search-Based Software Testing to Industry 175
  8.1 Introduction ............................................... 175
  8.2 Related Work ............................................. 176
  8.3 Context and Artifacts ................................... 178
  8.4 Static validation of the latest ISBST update .......... 187
  8.5 Lessons Learned .......................................... 200
  8.6 Threats to validity ...................................... 205
  8.7 Discussion ............................................... 208
  8.8 Conclusions .............................................. 208

Bibliography 210
Abstract

Context. Software is increasingly included as a component of a variety of systems and products across different domains. The ubiquity of software components means that software is often developed for particular domains and products, and its development and quality assurance processes need to fit the context for which they are developed. Thus, domain knowledge is crucial in developing, integrating, and ensuring the quality of such software components. In this context, a good software testing tool would have to be flexible enough to support a variety of software components in a domain, as well as being usable for engineers whose expertise is more focused on their domain than on software development and testing.

Search-Based Software Testing (SBST) promises to be such a flexible approach to generating and evaluating large numbers of test cases at minimal cost. Currently, SBST requires specialized knowledge to appropriately formulate the problem, to implement, and to use, as shown by the general lack of studies showing the application of SBST to industry. The need for specialized knowledge presents an obstacle that prevents more companies from trying, developing, evaluating, and adopting appropriate SBST systems.

Objective. The objective of this thesis is to develop an SBST system that can be applied in industry, evaluate that system in an industrial context, and reflect on the process of transferring this technology to industry.

Method. The thesis combines a series of research methods, both qualitative and quantitative, within the framework of Design Science Research for information systems. A prototype system was developed, evaluated in industry, and updated, as a means to investigate the applicability of the technique and to assess specific problems related to the development and transfer of SBST to industry. To benefit from both the SBST component and the experience of domain specialists, interaction was a key component of the prototype system.
Case studies and quasi-experiments were used to assess the prototype in an industrial setting, and controlled experiments were used to further investigate the findings and support our conclusions.

**Results.** An Interactive Search-Based Software Testing (ISBST) system was developed, evaluated in academia, and validated in industry. Interaction was found to be essential in enabling domain specialists to contribute their knowledge and experience to guide the search process. The ISBST system was found to provide a useful complement to existing software testing techniques, by investigating areas of system behavior that are not covered by existing testing methods. In addition, results show the importance of a well thought-out and complete transfer process, from communication with the industrial partner, to the development of appropriate mechanisms to evaluate and validate the ISBST system within the specific industrial context.

**Conclusions.** We conclude that ISBST can be applied in industry, and that there is evidence to show it provides benefits to the software testing process. In addition, the process of transferring ISBST to industry has been investigated within the framework of the Technology Transfer Model (TTM), and the additional requirements such a transfer places on the proposed solution have been discussed.

**Keywords:** Interactive Search-Based Software Testing, Search-Based Software Testing, Search-Based Software Engineering, Industrial Validation
Acknowledgements

First of all, I would like to express my gratitude towards my supervisors and mentors, Prof. Robert Feldt, Prof. Richard Torkar, and Dr. Simon Poulding for their patient guidance, timely advice and frequent critique. Robert has taught me a vision of research as a continuous quest to learn more about the world, and a philosophy that has greatly influenced my approach to my work. Richard has shown me the importance of priorities, and how a timely answer can both encourage and motivate. Simon always provided patient and kind help, even when, as it happened more than once, I gave him reason to be annoyed. I have been inspired by the ideas we discussed and by the vision of research that you showed me. As always, I hope to continue improving and learning from newer mistakes. I am grateful to all of you.

I would also like to thank my colleagues at SERL-Sweden, for creating a friendly and open work environment. From coffee chats to Swedish lessons, from research collaborations to teaching discussions, my day was brighter and more interesting, and for that I thank you all.

My work would not have been possible without support from my industrial partners. In particular, I would like to thank Fredrik and Johan, for their time, energy, and support. My appreciation goes out to the engineers that patiently tried out my ideas and allowed me to learn from their experience.

To many friends, who welcome me back into their lives every time I visit, and who offer their kind words and their time, even though I am far away. And last, but certainly not least, I would like to express my sincere gratitude to my parents, for their continuous and unconditional support.

This research has been supported by funding from The Knowledge Foundation (KKS) in the project Next Generation Software Engineering (NGSE), project no. 2010/0124.
Overview of Papers


Chapter 8: Bogdan Marculescu, Robert Feldt, Richard Torkar, Simon Poulding, “Transferring Interactive Search-Based Software Testing to Industry” in submission to a journal, 2017
Chapter 1

Introduction

1.1 Overview

Software is present to an increasing extent in a variety of different systems and products across different domains. Often, this takes the form of embedded software, where the software itself is just one component in a product. There are examples of embedded software being used across industrial areas [1], from automotive technology, to aerospace technology, defence, etc [2, 3].

The development and quality assurance of embedded software also needs to adapt to the domain. The limitations of the hardware available will have an impact on the software component. Some software intensive products, e.g. aerospace or automotive applications, may have very strict quality assurance standards and may require specific processes to ensure compliance. As a result, software is increasingly being developed by domain or systems engineers; people from a variety of educational backgrounds and styles [4], and whose training and experience is focused more on understanding the context and domain, and less on software development and software testing. In the following we will refer to such engineers as “domain specialists” to emphasize the crucial role of their domain-relevant knowledge and skills, and the comparatively secondary nature of their software development role.

Software testing is a necessary activity, but often considered time consuming: often thought to take up as much as half of the cost of a project [5]. While the exact effort is up for debate, software testing in an important part of the quality assurance process and often resource intensive. In a context where software is
not the primary component of a product, resources for software testing may become scarce. Since many embedded systems are subject to tight cost restrictions [2, 6] it may become prohibitively costly for companies to adapt existing software testing techniques, develop their own software testing techniques, or even assess the extent to which software testing is done and if there is a need for improvement.

Quality assurance, in this scenario, takes the form of system testing and acceptance testing, evaluating the overall system at the end of the development cycle. This is reasonable, since system quality is dependent on the quality of all the components, not just that of the software. Nevertheless, late discovery of software faults leads to increase cost as components have to be reworked. Delays and rework due to faults, however minor, are likely to propagate to other components as well, further increasing costs and delays. Moreover, embedded software can often be found in systems that are subject to life critical or mission critical safety rules. The challenge is therefore ensuring a high level of software quality, in a complex context that requires both domain and software testing knowledge, while ensuring that cost is kept at an acceptable level.

Search-based software engineering is a term that describes the application of metaheuristic techniques to problems in software engineering [7]. For the particular case of testing, these approaches are described as Search-Based Software Testing (SBST) [8, 9].

Ideally, a software testing solution for a particular domain would combine good practices of software testing with domain knowledge and experience in application-specific quality assurance procedures and regulations. SBST has been proposed and validated in academia for a range of different applications. There are, however, few examples of SBST systems being applied in industry, and transferred to industrial use. This thesis proposes a solution that aims to provide domain specialists with a way to use techniques that are on the cutting edge of software testing research without requiring additional training in a different area. To achieve this effect, the solution proposed in this work relies on the interaction between the domain specialist and the SBST system, based on the concept of Interactive Evolutionary Computation (IEC) described by Takagi [10]. The solution would consist of the application of SBST, in an interactive form, and transferring the resulting tool in an industrial context. This solution is called the Interactive Search-Based Software Testing (ISBST) system. The thesis presents the development, validation in industry and academia, and lessons learned regarding the use and deployment of the ISBST tool.

In the remaining pages of this chapter, Section 1.2 provides a brief description search-based software testing, as well as a description and analysis of the
1.2 Context

This section will provide a brief overview of the context of this work. First, a brief overview of search-based software engineering, its basic concepts and its application to the problem of software testing. Second, this section will provide an overview of the industrial context for this work, including a description of our industrial partner, their domain, and the problems they encountered.

1.2.1 Search-Based Software Testing

The term search-based software engineering (SBSE) was coined by Harman and Jones [7] to describe the application of metaheuristic techniques [11, 12, 13] to problems in software engineering. The application of SBSE to the problem of software testing is referred to as Search-Based Software Testing (SBST). SBST has been applied on a variety of software testing problems [8, 9].

The exact details of a SBST algorithm may vary, but Figure 1.1 illustrates the basic ideas of SBST using an example of a population-based genetic algorithm.

The algorithm shown in Figure 1.1 consists of several steps.

1. **Initialization.** First, an initial population of candidate solutions is generated. This is often done randomly, but more advanced techniques can also be applied.

2. **Fitness Evaluation.** Each candidate solution in the population is evaluated using a Fitness Function. The Fitness Function assigns a numerical value to each candidate solution and allows comparison between complex candidates.
3. **Selection.** From the original population of candidate solutions, a subset is selected for use in the next generation. The selection favors candidate solutions with better fitness, but other candidates also have a likelihood of being selected.

4. **Population Generation.** The selected candidates form the basis for the creation of a new generation. This is done by applying genetic operators. Two of examples of such genetic operators are mutation and crossover. Mutation involves a random modification being made to a candidate solution. Crossover involved the creation of new candidate solutions by combining existing ones.

For SBST, an individual candidate solution could be a test run trace, a test case, or a set of test data. The genetic algorithm will select test cases that perform better with respect to the quality criteria encoded in the fitness function. Over several generations, the overall fitness of the candidate population is expected to improve.

### 1.2.2 Industrial Context

There is research around developing new SBST techniques, validating them, and developing tools to support them. These evaluations are often conducted with industrial code [14, 15, 16], but not in active projects. Work by Wegener and Grochtmann [17] concludes that evolutionary algorithms show considerable promise, but that further work is required to find suitable parameters for practical application. Vos et al. [18] state that “evolutionary testing techniques have
hardly been applied to real-world complex systems”, but also add that industrial partners “responded positively” with regard to the efficiency of the tool. This work also identifies some issues regarding the application of SBST in industry, from fragile and incomplete prototypes [19] to the difficulty of designing a suitable fitness function [18].

Wide scale industrial application of SBST is unlikely before the issues of practical application have been resolved. Garousi and Mäntylä identify, among others, the test tool itself as a factor in the decision of companies to adopt automated testing [20]. In addition, confidence in the capabilities and performance of SBST outside of the laboratory is needed to encourage industry decision makers to try to implement such techniques.

Further application of SBST in an industrial setting would allow for more information about its capabilities, suitability, and performance. Such application would also generate the confidence in the potential of SBST that is needed to encourage wide scale adoption. Existing research claims promise and potential for SBST, but application is industry is required to fully realize that potential.

Embedded software offers a number of benefits for SBST application in industry. First, quality assurance is a critical factor in the development of embedded software. As a result, techniques that allow industry to improve the quality of the software testing process, without an unreasonable increase in the associated cost, are likely to be well received.

Second, while the quality of embedded software is crucial, the software itself is one part of a larger product. Thus, resources for software testing are likely to be limited. Moreover, faults in embedded software are likely to have a significant impact on other components, and therefore a significant impact on the cost of the project as a whole. Automated testing would lead to early discovery of software faults can result in lower costs, since those faults can be addressed before they affect other components [21].

1.3 Our industrial partner

Our industrial partner for the studies presented in this thesis is a company developing hardware and software used in control systems for off-highway vehicles. These are hardware systems that have an important software component, though that component does not receive the highest priority. The competitive edge that the company seeks with these products is more likely to revolve around domain-specific capabilities and enhanced core competencies, than from improved software development.
Our industrial partner also develops an embedded software development environment aimed at enabling domain specialists to develop and modify software for their own applications. The tool allows the user to select the type of hardware controller that will be used and then enables them to access the input and output signals of that controller. Standard modules can be combined and connected to the inputs and outputs to described desired behaviors. Basic functions and elementary operations are also provided, to allow users to define their own modules. The complexity of the user defined modules can range from basic components to large control systems with multiple sensors and controllers.

Once the model is complete, the tool verifies that the model is syntactically correct, and generates the code that is deployed on the hardware. The software can also be tested manually, by means of a simulator, or tested after being deployed on the hardware. Testing after deployment is done by means of specialized testing rigs and software environment simulators.

The development environment allows our industrial partner’s clients to use concepts they are already familiar with, e.g. working with signals and filters, to develop embedded software that is then deployed to specific hardware platforms. This means that developing embedded software for the hardware platforms in question is more accessible, since it does not require the developers to acquire a completely new set of skills. It also means, however, that introducing testing concepts into the process is a more complex task.

Testing that the resulting embedded software is appropriate semantically, and not just syntactically, is a resource-consuming task. Typically, the domain specialist that developed a new module is also in charge of creating the test cases, manually specifying inputs and observing the outputs.

Both the selection of the inputs and making the decision of whether or not the outputs conform to expectations depend heavily on the domain specialists’ experience and knowledge of the module they are developing. Experience allows domain specialists to focus on types of inputs that they expect will result in faulty behavior in the module, or in its components. Samples of typical expected values in normal use can also be included.

There is, however, a downside of basing test procedures on previous experience. First, the experience of individual domain specialists may vary, leading to the creation of modules with varying degrees of quality. A more systematic method of testing would decrease the level of uncertainty regarding the degree to which a module was tested.

Secondly, the developed module is tested by domain specialists that have to thoroughly understand the system behavior. This means that, when developing the test cases, they will focus on types of inputs known to cause problems, based
on their experience, previous cases, and knowledge of the module in question; this means, in most cases, a focus on limit cases and extreme values. If a new fault was introduced or an undesired behavior exists outside those types of inputs, this process is unlikely to catch those faults.

An automated means of developing test cases would solve some of these problems. By providing a more systematic way of developing test cases, the behavior of the module could be described with more confidence. In addition, automation would allow an increase in the number of tests, and thus remove the need to focus solely on known problem areas. This would increase the chance of identifying undesired behavior and rare faults.

An attempt to develop an automated testing system, on the other hand, is faced with a different set of problems. To allow such a system to quickly and effectively check a large number of test cases, it must have access to an oracle, i.e. a model that describes the desired behavior of the module being tested, which can clearly decide if a set of outputs is acceptable or not. Moreover, the oracle would have to be developed anew for each context and each module, further increasing the difficulty and the resources needed for such a task. A domain specialist can, however, identify unwanted behavior, based on their understanding of the module itself, its intended application, and context limitations.

In conclusion, this type of context is one that would benefit from the large number of test cases and the systematic approach of an automated system. An automated system that is difficult and costly to develop and requires software expertise to properly use is a poor fit for this context, however. In short, the domain specialists that use the embedded software development environment develop software, but are not software experts. Moreover, companies often focus their improvement efforts on core competencies, so resources available for improving the software testing process are often scarce and is likely to continue to be so.

The problem is to develop a software testing system that is flexible and robust enough to adapt to changes in context and testing needs, while, at the same time, accessible enough to allow domain specialists to use it without requiring them to develop additional skills or adapt to new domains. Developing models in order to then test the software being generated as a result of those models would be a complication, distracting domain specialists from their tasks.
1.4 Related Work

As presented above, the mechanism chosen to provide the flexibility and robustness required for the solution described above was Search Based Software Engineering. To make this mechanism more suitable for evaluation and transfer to industry, mechanisms to enable interaction between the domain specialists and the systems were included in the solution. In the following we present the key concepts and related work.

The term Search Based Software Engineering (SBSE) describes the application of metaheuristic techniques, e.g. genetic algorithms, evolutionary computation, ant hill optimization, to software engineering problems, and was coined in 2001, by Harman and Jones [7] even though similar ideas had been investigated earlier [11, 13, 22]. Search-based techniques have been applied to a variety of sub-areas of software engineering [23], including several types of testing problems [8, 9], object-oriented containers [24], and dynamic programming languages [25].

Due to the context described above, we are interested in including domain specialists in the search process, to allow our solution to benefit from their experience and knowledge. The inclusion of human judgement in search-based approaches is one of the potential directions for search-based software engineering that was identified in a study by Harman [26].

Takagi [10] describes the application of Interactive Evolutionary Computation (IEC) in different areas, from graphic art and music to industrial design and robotics. While software engineering is not directly included in that research, some parallels can be drawn.

As an example of interaction in search-based software engineering, Feldt [27] described an interactive development environment where tests are created as the engineer write the program code or refine the specification. The system used the interactions of the engineer to help guide the search but the effect on the fitness function was indirect and opaque. It was not clear for an engineer how to affect the fitness function in order to direct the search in a particular direction. This might limit usefulness.

Simons et al. [28, 29, 30, 31] describe a human designer involved in steering an evolutionary, multi-objective search, using elegance as a evaluation criterion. Transposed to the context of this paper, human involvement means allowing a human domain specialist to guide the search both by direct fitness evaluations and by allowing them to modify the fitness function of the search-based system.

One issue that arises when discussing potential interaction between a human and search-based system is that of human fatigue. Search-based systems use the fitness function to evaluate large numbers of potential solutions. This means
that involving a human in the search is not as straightforward as replacing the existing fitness function with the human domain specialist. Such an approach would give rise to fatigue, especially decision fatigue [26, 32].

Moreover, while the exact limits of human awareness and the conditions when such decision fatigue may come into play are not precisely determined, some limit exists to the human capacity to handle information over a short period of time [33, 34]. This makes overcoming the fatigue problem essential in allowing a human to provide useful guidance to a search-based system.

User-centered, or human-centered design, offers a way of potentially addressing many of the problems related to user interaction. In the context of this paper, the focus of the user centered approach [35] is to involve the domain specialists that will use the search-based software testing system in the design, to understand how the they will interact with the system, what their needs are and what functions are best left to the human specialist and which are to be taken over by the system.

Looking at existing work by Takagi [10] and Bush [36], several approaches can be found to allow interaction between a user and a search-based system.

The most involved is that of hyper-interactivity, defined as a “form of IEC in which a human user actively chooses when and how to apply each of the available evolutionary operators, playing the central role in the control flow of evolutionary search processes” [36]. Here, the human acts as a direct guiding hand into each individual’s development rather than as a substitute fitness function, evaluating individuals \textit{a posteriori}.

One approach by Takagi [10] is that of having the human act as a regular fitness function, i.e. they analyze each of the candidate solutions and assign a fitness score to it.

More general still is to present the human with the individuals to be evaluated; the human then chooses those that are remarkable, either selecting the ‘good’ individuals for promotion to the next generation or selecting the ‘bad’ ones for exclusion. This helps guide the search by ensuring that desired characteristics are always represented in the population and have a higher chance of propagating to the next generation. In effect, the user guides the search by selecting those individuals deemed to be the “best current representation of the goal” [37].

A final option is that of Visualized EC, where the human selects a solution based on the fitness values for several objectives, rather than analyzing the individuals themselves. (The approach is described in more detail in [38].) One such example is presented by Bavota et al. [39] where an individual candidate is one proposed distribution of software components into clusters. Rather than
evaluating each candidate itself, the user is required to decide if two components belong in the same cluster or not.

An interesting parallel can be made with EvoSuite, a search-based software testing (SBST) tool [40] that aims to produce test cases that achieve high coverage for Java programs. While it uses search-based techniques to achieve a similar scope, i.e. generating test cases, there are significant differences between the context that EvoSuite is meant to support and the one presented here. EvoSuite uses high coverage as a primary means of evaluating the suitability of test cases and employs a white-box approach to testing, both assumptions in direct contradiction with the limitations of the context of this thesis. The main limitation of EvoSuite, however, is that it is only aimed at Java programs, and thus is unsuitable for the context being considered in this work.

An example of interaction being used to guide an IEC is PicBreeder, a service that uses input from online users to evolve images [41]. The user of PicBreeder selects a picture that is aesthetically pleasing to them and guides the system to use that image as a basis for further evolution. The system can be further guided by allowing the user to choose to focus on ‘big’ or ‘small’ changes. While this is not applied on a software engineering problem, it is an interesting practical example of an IEC which has gathered a lot of attention and interest.

Alternatives to objective-focused, competitive algorithms have also been proposed. These non-competitive, or exploration-focused, algorithms could enable a testing tool to explore the search space without focusing on a particular objective or set of objectives. Examples of exploration-focused algorithms are: Novelty Search [42] - where the fitness function consists of the “novelty” of the proposed candidate; Viability Evolution [43] - a space of “viable” candidates is defined and the system seeks to develop candidates that are viable by that definition; and MAP-Elites or Illumination Search [44] - divides the output space into cells and forces exploration by trying to fill each of the cells with at least one appropriate candidate. The exploration-focused algorithms seems to provide a useful fitness function when objectives are unclear or unknown.

An interesting issue is that of validating, and ultimately transferring, SBST to industry. Efforts have been made to validate SBST with industrial code. EvoSuite, mentioned above, has been evaluated on open-source projects by Fraser and Arcuri [14] and on industrial code by Campos et al. [15]. Fraser and Arcuri [45] also discuss the difficulties associated with developing a prototype that is reliable enough to evaluate on open-source software, and later in the real-world. Bauersfeld et al. present the evaluation of TESTAR [46] with two industrial practitioners, and provide some reflections on the transfer process. An example of evaluating a hill climbing algorithm with industrial code is that of
Doganay et al. [16], while Enoiu et al. [47] use industrial code as a basis for their experiment. It is worth noting, however, that these evaluations are either conducted in an academic setting, often with students as participants, or by researchers using industrial code. To successfully evaluate and deploy an SBST tool to industry, evaluations should include industrial practitioners, and ultimately be concerned with how a tool should be handed over for use in active projects.

A model for the transfer of technology to industry is proposed by Gorschek et al. [48]. The Technology Transfer Model (TTM), seen in Figure 1.2, is a seven step iterative model, based on the authors’ empirical experiences of collaborating with industry and validated with industrial partners. The seven steps are: 1) **Problem Identification.** This step focuses on understanding the context of the industrial partner that will be the beneficiary of the technology transfer project. Understanding the domain, establishing a common terminology, understanding and prioritizing the needs of the industrial partner are identified as key issues at this step. 2) **Formulate a research agenda.** Based on the needs identified and prioritized at the previous step, researchers formulate an agenda for their work, in close cooperation with their industry contacts. 3) **Formulate a candidate solution.** A candidate solution is developed for the context, or adapted to fit the context. 4) **Validation in Academia.** Once the solution is developed, it is validated in a laboratory setting. 5) **Static Validation.** Static validation consists in having practitioners evaluate the candidate solution, providing feedback to further improve the candidate solution. This type of evaluation takes place in an industrial setting and uses industrial artifacts, but is not carried out in an active project. 6) **Dynamic Validation.** Dynamic validation consists in evaluating the candidate solution as a pilot in industry. This step is aimed at further improving the solution and indicating what is needed for the full scale transfer. The dynamic validation is carried out as part of an active pilot project. 7) **Release the Solution.** This step involves delivery of the candidate solution to industry, along with documentation and reference guides, training support, and measurement programs.

The TTM forms a useful framework for discussing the transfer of SBST in an industrial setting. The evolution of the candidate solution we adopted for the transfer of ISBST in industry will be discussed in the following sections.
1.5 The Interactive Search-Based Software Testing System

This section describes in brief the components of the ISBST system, in its latest and most complete form, and how they relate to SBSE in general. This is to allow us to more clearly discuss the individual studies, refinements, and results lated in this introduction.

The ISBST tool is a search-based software testing tool meant to allow domain specialists to use their knowledge and experience to guide the search. This guidance is achieved by allowing the domain specialist to change the fitness function guiding the search, and then assess the resulting test cases to further improve their definition of the fitness function. The fitness function is composed of a number of criteria, called search objectives, that measure characteristics of the SUT. The domain specialist guides the search by deciding on the relative importance of these objectives.

The ISBST system has two nested components: an SBST system that connects to the SUT forms the inner cycle, and the outer cycle handles the interaction between the inner SBST system and the domain specialist. An overview of the ISBST system can be seen in Figure 1.3.
The **Inner Cycle** consists of the search algorithm itself, the fitness function and the search objectives that guides it, and the mechanism that handle the interaction with the SUT. The algorithm used is a differential evolution algorithm [49] that generates a set number of test case inputs, that are then used to run the SUT and obtain the corresponding behavior. Each test case input consists of a vector of real numbers. The combination of inputs and behavior are referred to collectively as a candidate. Once the behavior has been recorded, the candidate is assessed using the fitness function.

The mutation strategy for new candidates is as follows:

\[ v_{i,G+1} = x_{r_1,G} + F \times (x_{r_2,G} - x_{r_3,G}) \]  

(1.1)

where \( r_1, r_2, r_3 \in 1, 2, \ldots, NP \), are integers, and mutually different, and different from the running index \( i \). \( F \) is a real and constant factor \( \in (0, 2] \) which controls the amplification of the differential variation \( (x_{r_2,G} - x_{r_3,G}) \). If the mutant vector is an improvement over the target vector, it replaces it in the following generation [49].
The fitness function is made up of several search objectives assessed independently. The results of each of these assessments are collected and combined according to Bentley’s Sum of Weighted Global Ratios [50], as can be seen below:

$$\text{DAFF}_j = \sum_{i=1}^{n\text{Objectives}} \text{Weight}_i \times \text{Value}_{i,j}$$

(1.2)

where $\text{DAFF}_j$ is the fitness value of candidate $j$, $\text{Weight}_i$ is the current weight of the objective $i$, and $\text{Value}_{i,j}$ is the fitness value of candidate $j$ measured by objective $i$. The value of $\text{DAFF}_j$ is the sum of the weighted fitness values for all $n\text{Objectives}$ objectives. An objective $k$ can be deselected from the computation by having $\text{Weight}_k = 0$.

The **Outer Cycle** is a shell around the SBST system that allows domain specialists to interact with the SBST by adjusting the relative importance of each search objective and to view the resulting candidates. The candidates resulting from the search are displayed as a group, relative to the fitness values they received. Each individual candidate can be displayed in more detail, if a domain specialist deems it useful. The search interaction is conducted by allowing the domain specialist to set the relative weights for each search objective. The weights are then passed to the Inner Cycle, where they are part of the fitness evaluation.

At the moment, new search objectives can only be added by hand, with the code for the fitness evaluation being added to the appropriate module. Once the code is written, however, the new search objectives are automatically used for future fitness evaluations. However, experience has shown that any set of search objective that is pre-defined is unlikely to be complete, so a means of allowing new objectives to be added would be useful for practical deployment and further evaluation.

### 1.6 Research Questions

Search-based techniques have been proposed and validated in academia, and with industrial code. But transfer of such techniques to industry is still not common. The goal of this work is to investigate the viability of search-based techniques to the industrial context described above, to implement a tool that allows these techniques to be transferred and evaluated in industry, and to develop the mechanisms for evaluating the tool itself and the transfer to industry.

This goal can be further divided into three general research questions:
• Development: How can search-based software testing be developed for use in industry?

SBST systems that have been evaluated in academia are prototypes and proof-of-concept systems that are meant to be used by researchers. An initial concern was how search-based software testing techniques can be packaged in a tool that could be deployed in an industrial context, and used by domain specialists. To answer this question it was necessary to design, develop, and evaluate such a tool. Additional questions also arose as a result of these efforts. Communication between an SBST system and a domain specialist is a key element in ensuring that the tool is useful and appropriate. This involves the domain specialist communicating their experience and their aims to the SBST tool, and the tool communicating the progress of the search, and the resulting solutions, to the domain specialist in a way that is clear and meaningful.

• Deployment: How can an SBST/ISBST tool be deployed to industry?

Search-based techniques have been evaluated in academia and on industrial code, but transferring tool that are based on SBST to industry is still to be accomplished. The issue of how such a tool can be deployed to industry has a significant impact on how the tool is developed, what type of evaluation it is subjected to, and how additional information is used. To allow an SBST/ISBST tool to be deployed, non-functional issues like flexibility, usability, and robustness, are added to the usual, functional, concerns. Moreover, deployment is likely to be a multi-step process, allowing for incremental development and for the incorporation of lessons learned into the tool being developed and deployed.

• Evaluation: How can an SBST/ISBST tool be evaluated and validated throughout the deployment steps?

Evaluation is a critical aspect in developing, and later deploying, an SBST/ISBST tool. The types of evaluation that can be performed differ based on the context and the maturity of the tool, and on the context that it is aimed at.

General evaluation approaches are suitable for the laboratory setting and focus on the functional properties of the tool. The quality of the solutions being developed and the diversity of those solutions provide an evaluation approach in a controlled setting.
When transferring a tool into an industrial setting, non-functional aspects become increasingly important. We have already mentioned robustness, usability, and the ability of the tool to communicate the solutions in a clear and meaningful way. Such a tool would also have to be reasonably mature before being transferred to industry. An important issue, therefore, is how this maturity can be assessed both when deploying such tools in industry, but also when developing them. Ensuring that the evaluation a tool received during the development stage is relevant to the industrial context could have a significant influence on how the tool is perceived.

The tool’s ability to find faults, as well as a classification of faults it finds, can be assessed in both industry and academia. However, it must be noted that the ability to find faults in a controlled, laboratory setting may differ from fault finding in more realistic conditions. Since an industrial setting is less likely to offer ideal conditions, it is important to assess how evaluation procedures transfer between the academic and industrial settings.

In addition, any tool deployed in industry would also have to fit in the quality assurance process, and the tool chain, that the adopter company uses. In addition to developing the ISBST tool, a key element is developing and validating types of evaluation that can be used in similar projects.

Developing an ISBST tool, along with defining approaches for evaluations, and relating to a framework enabling its successful transfer to industry would provide an example of how these questions can be answered, along with what difficulties are likely to be faced, and how such difficulties may be overcome.

Successful development and transfer of a prototype ISBST system would be a first step for a number of different directions of software engineering research. First, as an first example, it would be limited in terms of generalizability and bound by context specific choices. Using this example, additional transfer projects would allow researchers to better identify the contexts where such approaches could be helpful, as well as their limitations. The prototype transfer would also raise additional questions, relating to integration with the industrial process, what other tools, people, and artifacts would such a system have to interact with, or how information would best be presented to the domain specialists. A successful first transfer would also enable longitudinal studies to be performed, bringing to light additional effects that are difficult to theorize in the absence of practical experience.
1.7 Research Methods

The choice of research method should consider the most appropriate way of answering the questions above, in the context of our industrial partner. One way to answer the research questions above is to develop a tool that uses SBST techniques, evaluate it and validate it in academia and industry, and assess the potential for deploying it in a practical setting. Therefore, the work presented in this thesis was conducted under the framework of design science research.

The concept of design science was formulated as early as 1995, by March and Smith [51]. In 2004, Hevner et al. developed a framework and clear guidelines for the application of design science in Information Systems [52, 53].

The approach taken for this thesis matches those guidelines [52, 53]. A viable artifact is produced to provide a technology-based solution to a relevant industrial problem. Design evaluation is performed through a series of four case studies, and research contributions are provided in the areas of the design artifact and of the design methodology. The artifact was rigorously validated and the results of these validations influenced its further development. The efforts presented in this thesis were aimed at developing an effective artifact, utilizing available means and satisfying laws in the problem environment. Finally, the research is presented both to technology-oriented audiences, via scientific publications, and to management-oriented ones, via workshops with our industrial partner.

For the purposes of the work presented here, the artifact in question is the ISBST tool, incorporating all the information that emerged from the case studies and allowing for a practical, albeit static, validation of the obtained information. Each additional case study provided new information, as well as potentially new questions to be investigated. The constant validation of both the emerging information and the new questions was a means of ensuring that each case study had relevant goals and was built on accurate information.

A closer look at the Design Evaluation Methods [52, 53] also shows conformance with the framework. The current study includes four case studies, aimed at investigating phenomena in their context [54]. Static analysis was performed, as described by Gorschek et al [48] and mentioned above, and the artifact itself focuses on functional, i.e. black-box, testing. In the context of this framework, the artifact is the Interactive Search-Based Software Testing (ISBST) tool.

Initial stages of the work presented in this thesis were exploratory in nature, with a focus on obtaining as much information as possible about the context, the systems under test (SUT), and the needs of the company and the domain specialists. Thus, the initial stages consisted of several case studies, focusing
on obtaining as deep an understanding as possible from a limited number of domain specialists.

The main sources of information were those common to all the case studies. Semi-structured and unstructured interviews were conducted with the various stakeholders, i.e. developers, engineers and managers. Observations provided information regarding the current processes, the training received by engineers, and the way engineers interact with the proposed solution. Multiple sources of information allowed for triangulation of data sources and data collection methods.

The use of a series of case studies created a constant validation loop, with new information being uncovered, validated with the relevant stakeholders, and integrated into the proposed solution. Finally, the ISBST system as a whole was validated against stakeholders’ expectations. This approach allowed for a deeper understanding of the behaviors being observed and the reasoning behind those behaviors. It also provided a greater degree of confidence in the relevance and accuracy of the results.

The information obtained from the case studies made other research methods viable choices. Controlled experiments were conducted to assess specific aspects of the ISBST tool. One experiment was conducted with human subjects, and totaled 58 participants. Two quasi-experiments in industry were conducted to assess validity in an industrial setting. Laboratory experiments were conducted to support other studies and to assess components of the ISBST tool that relied less on human interaction. Experimentation with human subjects provided initial information on usability and understandability, while desk experiments
1.8 Summary of Papers Included in the Thesis

This section contains a brief summary of the chapters and details how they relate to each other and the main topic.

1.8.1 Paper P1: A Concept for an Interactive Search-Based Software Testing System

The first paper describes how a search-based software testing system can be designed to allow for interaction with its users, without requiring the latter to become proficient in software engineering.

This was the first step in applying search-based software testing in the specific context and to the needs of our industrial partner. The concept proposed in this paper is designed to enable this application, by proposing a novel means of interaction with the user, in this case a domain specialist. To allow the user to make full use of their experience and intuition without requiring any them
to acquire additional skills, the proposed system introduces the notion of an Interactive Fitness Function (IFF). This is a fitness function that dynamically changes to match the user’s understanding of the problem. As the user, henceforth called ‘domain specialist’, gains further insights into the nature of the problem, the desired solution also changes. The flexibility introduced by the IFF allows these changes in the desired solution to be communicated to the system, by means of changes in the search objectives, and therefore enables the system to continue the search in a relevant direction.

At the time of publication, the concept proposed in this paper was validated by means of workshops with engineers from our partner company. The results of this validation were encouraging enough to allow for the development, and later validation, of the prototype ISBST system.

Since that first study, the Interactive Search-Based Software Testing system has been constantly refined. It is a practical implementation of the concepts presented in this paper and follows roughly the design presented here. Any differences that do exist are due to new information becoming available, either due to the need to solve practical problems or as a result of the constant validation efforts.

1.8.2 Paper P2: Objective Re-Weighting to Guide an Interactive Search-Based Software Testing System

This paper focuses on evaluating an early version of the Interactive Search-Based Software Testing (ISBST) system, focusing in particular on the issue of using weighted search objectives to form the Interactive Fitness Function (IFF) described above. The study was focused around two research questions. The first, “How can interaction between a domain specialist and the ISBST system be achieved?” was answered by proposing our mechanism for building the IFF from existing search objectives. The interaction was deemed successful if the domain specialist was able to use the search objective, and thus build the IFF without the need for in-depth knowledge of search-based software engineering. The second research question was aimed at evaluating the impact that the interaction would have on solution diversity.

The evaluation focuses on the Interactive Fitness Function (IFF) mechanism and the means by which this function is created. The ISBST system dynamically creates the IFF through the composition of a number of search objectives. The domain specialist adjusts the weights of each objective to match their current goal. The IFF is then modified accordingly and used to guide the search further.
The adoption of a new type of fitness function, however, had to be validated before being put into practice. This was done to ensure that the changes that were done to the basic search-based approach did not negatively impact its capabilities. The key success factor was preserving the diversity of the candidate solutions throughout various interaction scenarios. The evaluation showed that certain patterns of interaction have a negative impact on diversity. Nevertheless, the impact is not of a sufficient magnitude to threaten the usefulness of the mechanism of interaction between the ISBST system and the user by means of objective re-weighting.

This paper also highlights the process of constant validation that the ISBST system underwent during development. Validation efforts ranged from sanity checks during discussions and interviews, workshops with company engineers, to practical evaluations of system functionality and of the ISBST system as a whole. In addition to confidence in the validity of the concept and the system, these efforts also provided a constant stream of improvements and potential improvement ideas.

The paper concludes that objective re-weighting is a viable approach to building the IFF without sacrificing solution diversity, and that familiarity with SBSE is not a prerequisite for using the ISBST system. Moreover, the study also results in a set of interaction strategies that could be used in the laboratory, to validate future version of the ISBST system. While these strategies are not a true substitute for the domain specialists, they are useful tools in the development and academic validation of the ISBST tool.

1.8.3 Paper P3: Practitioner-Oriented Visualization in an Interactive Search-Based Software Test Case Creation Tool

This paper focuses on evaluating the interaction component of an early version of the Interactive Search-Based Software Testing (ISBST) system from an industrial perspective. The main research question in this study was: “To what extent are the interaction mechanisms proposed for the ISBST system usable by the domain specialists?” By answering this questions, we could also show that, at least in the industrial context we focused on, the Interactive SBST approach was applicable.

To answer this question we conducted an early validation of the ISBST system’s interaction interface, focusing on information communication, the se-
lection of search objectives for the IFF, and interviews to assess the perception of domain specialists towards the interaction.

The evaluation focused on the successful interaction between the domain specialists participating in the evaluation and the prototype system. The interaction approach was considered successful because it achieved two major goals. First, domain specialists with limited training in search-based concepts were able to interact with the system, understand the solutions it offered, and were able to use the search objectives to adjust the search to their goals. Secondly, the interaction between the domain specialists and the system was shown to have a clear effect in the solutions that were obtained. Thus, this showed that successful interaction between the ISBST system and the domain specialists can be achieved and that this interaction does indeed lead to different solutions, more suited to the expectations of the domain specialists.

The results presented in this paper provided early validation for the interaction mechanisms chosen for the ISBST system.

1.8.4 Paper P4: An Initial Industrial Evaluation of Interactive Search-Based Testing for Embedded Software

This paper describes an evaluation of the ISBST system with domain specialists from our industrial partner. The research questions in this study were focused on the domain specialist, their interaction with the ISBST system and their perception of the system. The research questions were: “What is the domain specialists’ evaluation of the ISBST system in terms of usefulness and usability?” and “How effective is the interaction between the domain specialists and the ISBST system?”

To answer these questions, the study consisted of a hands-on experimental evaluation, carried out by domain specialists in an industrial setting, using industrial code. This evaluation targeted the ISBST system itself, by having the resulting candidate solutions evaluated by domain experts. Improvements in the solutions being offered provide practical evidence that the underlying search-based system evolves candidates as expected and provides validation of the mechanism chosen to guide the search. The successful interaction, complemented by suggestions for further improvement, validate the approach taken to handle the interaction with the domain specialist. Interaction-specific concerns proved to be seldom investigated and considered, but nevertheless vital elements to successfully applying a tool in a practical setting.

In addition, the evaluation was also concerned with integrating the ISBST system in the current quality assurance process that the company and their
1.8 Summary of Papers Included in the Thesis

clients use. Proper integration with the existing process is a key element in gaining the acceptance and support from the domain specialists and, therefore, a major concern when developing any new method or tool.

The study concluded that the ISBST system is usable by domain specialists, and that it is perceived to be a useful complement to existing techniques. In addition, the evaluation also resulted in significant feedback regarding the interaction between domain specialists and the ISBST system, and suggestions for improvement. The interaction between the domain specialists and the ISBST system was found to be effective and to allow domain specialists to understand the test cases developed by the system and successfully guide the ISBST system. The study also concludes that guided search, compared to unguided search, results in better scores for the prioritized objectives. This is hardly surprising, but it confirms the effect that the search objective weighting mechanism has on the outcome of the search.

This evaluation validated the concept of the ISBST system as a whole, further strengthening the evidence that the domain specialists can use their experience and intuition to successfully guide a search-based system towards useful solutions.

1.8.5 Paper P5: Tester Interactivity makes a Difference in Search-Based Software Testing: A Controlled Experiment

This paper presents an experiment, conducted with students in a software engineering master’s program, aimed at comparing the ISBST system with a manual technique. The experiment showed that the ISBST system allows the same participants to develop test cases that exercise different system behaviors than the manual tests they used.

The research questions investigate three issues. The first research question is to determine if the test cases developed by the ISBST system investigate different areas of the behavior space than the ones that are developed manually. The second is meant to confirm that both the search and the interaction component have a significant impact on the results of the search. The third question looks at the degree to which the ISBST system is more demanding than manual test case development for the same SUTs.

To answer this question, the study consisted of an experiment with human subjects and a laboratory experiment. The results show that the ISBST system does develop test cases that investigate a different area of the SUT behavior.
space than manually developed test cases. This makes the ISBST tool a potentially useful complement to existing manual techniques.

In addition, the study also shows that both the interaction and the search have a significant impact on the results of the search. The interaction, driven by the participants by setting different weights, which match the priorities they assign to different search objectives, means that the search is less focused than if the weights of the objectives are known \textit{a priori}. However, the ISBST system is aimed at allowing domain specialists to develop test cases in situations where such detailed knowledge of the objectives is not available. While the ISBST system does not perform as well with the interaction as without, the results are still comparable.

The study also yielded useful information about the interaction from the perspective of the participants. All participants were able to use the system, but reported a higher level of mental demand and effort in using the ISBST system, as measured by the NASA-TLX assessment questionnaire. This could be due to the comparatively novel technique and a lack of experience with it, but it could also be due to a somewhat unintuitive interface and a sub-optimal approach to communicating the results of the search.

The work presented in this paper showed that the interaction component of the ISBST system has a significant effect on the outcome of the search. It also showed that the tool was usable to the participants, and pointed towards improvements in terms of the interface. On the higher level, it was also useful in identifying potential means of assessing novel tools in academia.

1.8.6 Paper P6: Using Exploration Focused Techniques to Augment Search-Based Software Testing: An Experimental Evaluation

The ISBST tool, as developed and assessed so far, relies on a set of search objectives that form the fitness function. Candidate solutions compete with each other, and are measured against that set of objectives, weighted according to the input received from the domain specialists. An alternative approach is that of exploration-focused techniques. Exploration focused techniques rely on candidate solutions being viable or novel, rather than competing according to a set of search objectives. Such techniques could be useful in situations where not enough information is available to develop a useful list of search objectives.

The research questions for the study presented in this paper were: first, to what extend to exploration-focused algorithms investigate a different area of the
behavior space than the objective-based ISBST tool; second, to investigate the degree to which exploration-focused algorithms provide benefits to candidate solution diversity when running with restricted resources.

The study consists of a laboratory experiment comparing several exploration focused techniques: four versions of Illumination Search, Novelty Search, and Viability Search. The results of this study show that the exploration-focused techniques do investigate a wider area of the behavior space than objective-based approaches. They also show that the increased diversity in the candidate solution population is present also when the exploration-focused algorithms are working in conditions of limited computational resources. The paper concludes that exploration-focused algorithms are a potentially useful complement to the ISBST system, since they can explore the behavior of a SUT in situations where little information is available.

Moreover, while exploration-focused techniques are not as effective in driving towards an optimal solution, they are quite good at exploring the behavior space even when working under a comparatively small computational budget. On this basis, we hypothesize that exploration-focused techniques could work well to complement the more optimization focused search-based software testing systems.

1.8.7 Paper P7: Transferring Interactive Search-Based Software Testing to Industry

The experimental evaluation of the ISBST tool in industry has provided a considerable amount of data and feedback regarding its usefulness, as well as suggestions for improvement. The updated tool has been evaluated experimentally in an academic setting, but further industrial evaluation is necessary to allow us to transfer the tool to industry and study it in an operational setting.

As a result, the updated ISBST tool was evaluated in an industrial setting, on-site and with domain specialists. One of the goals of this study was to assess the readiness of the ISBST tool for deployment in industry and to determine what steps, if any, were still necessary for that deployment. To provide a frame for the discussion on deployment readiness, we used the Technology Transfer Model proposed by Gorschek et al. [48].

The study focused around three research questions. First, does the ISBST system develop test cases that can identify the bugs injected in the SUT? The second, to what extent can domain specialists using the ISBST system develop test cases that can identify injected bugs in the SUT? These two research questions look at the difference between how the ISBST system operates in ideal
conditions compared against operation in a more realistic environment. The third research question focused on the degree to which the ISBST system communicates its findings clearly and meaningfully to the domain specialists. The goal was to see if the ISBST system allows domain specialists to quickly identify test cases that have a high fitness score for one or more of the objectives and allow them to identify why they have that score.

To answer these questions, we conducted an on-site experimental evaluation in industry, with domain specialists as participants. In addition, experiments in the laboratory were used to assess the ISBST tool under ideal conditions. The study concludes that the ISBST system was able to develop test cases that cause different behaviors between the SUT versions with injected bugs and the reference version, both in the laboratory and on-site, when used by domain specialists.

The results also show that the ISBST system could be improved in terms of the way candidate solutions are communicated to the domain specialists. Individual candidate solutions can be seen in detail, and domain specialists were able to understand individual test cases without complications. However, distinguishing interesting candidate solutions in the entire population has proven to be more challenging. The ISBST tool had difficulties in clearly showing the candidate populations, plotted across several search objectives. Types of display that showed only two of the search objectives caused interesting candidate solutions to be lost, while displays that show all the available data quickly become overwhelming.

1.9 Discussion

This section will discuss the overall approach and how each of the subsequent chapters fit into this approach. It will also provide a brief look at the limitations and validity threats of the work in its entirety. Note that discussions on the strengths, possibilities, and threats to validity specific to each study can be found in the respective chapters.

1.9.1 Overview

We will use as a baseline the research approach and technology transfer approach proposed by Gorschek et al. [48], and seen in Figure 1.2. An overview of the papers included in this study, their positioning with respect to industry and academia, and the technology transfer steps relevant to each can be found
1.9 Discussion

in Figure 1.5. The first study includes Steps 1 and 2: Problem identification in industry and the formulation of the research agenda, respectively. Problem identification relied on information from training sessions, interviews and literature and was validated in workshops with company stakeholders. This created the basic starting point for problem understanding and allowed for the development of the artifact to begin, which later would become the ISBST system.

Studies P2 and P3 were conducted simultaneously, but had different focal points. P2 was practitioner focused and aimed at statically validating the interaction component of the artifact. The validation was performed by means of workshops, interviews and observations. Software engineers and embedded software developers from our partner company participated in a workshop where they provided feedback on the artifact that had been developed: A partially functional ISBST system. In terms of the technology transfer approach, P2 comprised the loop between Step 3, formulating a candidate solution for interaction, and Step 4, an academic validation of that interaction solution. The actual validation was an intermediate step between academic and static validation: it was conducted with industry practitioners, in an industrial setting, but the prototype was only partially functional.

P3 was conducted at the same time and used the same artifact as P2. But the focus for P3 was investigating the viability of the search-based approach given the current implementation. This was more academically focused and involved less industrial input, aside from the regular sanity checks. This study comprised the loop between Steps 3 and 4, candidate solution formulation and validation in academia. Confirming the viability of the underlying search-based solution was a vital step in developing a fully functional prototype.

P4 was an overall evaluation of the first functional prototype, by means of a test, with industrial code and personnel from the company. Most of the information derived from observations of the interaction between the ISBST system and the software engineers and embedded software developers working with our industrial partner. Additional information was provided by semi-structured interviews with the participants in the test. From the point of view of the technology transfer approach, this study focused on Step 5—a static validation in an industrial setting, but not conducted on an active project. The study revealed a number of flaws and potential improvements of the ISBST prototype and pointed to practical problems with integrating the prototype in the testing process of the company.

As a result of P4, the ISBST system was redesigned and redeveloped, to improve performance, stability, and to enable an easier connection to the company’s artifacts, that did not require any manual instrumentation of SUT code.
Figure 1.5: The positioning of the papers with respect to the technology transfer model proposed by Gorschek et al. [48] (the steps of the model are referred to as RT1-3 etc. in this figure). On the vertical axis we have the positioning of the papers, from academically focused on the bottom towards industrial focus up on top.
1.9 Discussion

The second functional ISBST prototype system was validated in academia, with an open-source SUT, in an experiment. The experiment, presented in P5, was conducted at Blekinge Institute of Technology, with 60 participants from the Verification and Validation course in the master’s program. The goal of the experiment was to identify the degree to which tester interactivity influences the search, and to determine if the ISBST system and the manual methods investigate different behaviors for the SUT. In addition, the study was aimed at looking at the usability of the ISBST system, both in terms of the practical outcome, i.e. can participants use the system to develop tests for the given SUT, and in terms of the effect on the participants, i.e. measuring fatigue and the mental demand participants reported.

A parallel study is described in P6. It focused on academic validation as well, with a laboratory experiment aimed at assessing the ISBST system against exploration-focused techniques. The study concludes that exploration-focused techniques are better at exploring the behavior space of a given SUT, but less effective at converging on optimal solutions. This is expected, given the differences between the two categories of algorithms. The results suggest the possibility of using exploration-focused algorithms as alternatives, when not enough initial information about the behavior space existing or search objectives that measure the behavior space. Exploration-focused algorithms could also be included in
the ISBST tool, complementing existing optimization-focused algorithms. This would allow optimization to find ideal solutions, but run exploration-focused alternatives for a more better overview of the behavior space. This evaluation is classified as ‘Validation in Academia’, according to the Technology Transfer Model.

The final study, P7, had two purposes. The first was to conduct an additional validation, in an industrial setting, with industrial SUT code, and with industrial practitioners, of the updated ISBST system. The second purpose was to compile all the lessons learned and the information gathered regarding the development and deployment of the ISBST system and to hypothesize on how similar technology transfer project could benefit from the experience presented in this thesis.

1.9.2 Technology Transfer

The thesis describes a yet incomplete transfer of interactive search based software testing to industry, in the form of the ISBST system. Taking the Technology Transfer Model as a frame, the thesis takes the concept of interactive search-based software testing from an initial concept, to a working prototype that has received validation in academia and in industry.

A key aspect of technology transfer that is highlighted in the initial paper is that of constant on-site presence and early piloting of candidate solutions. Our experience supports the importance of constant assessment and constant validation from the industry partner, and the value of sanity checks that such presence and interaction allows.

The experience of this study, however, also points to a trade-off. The artifacts that are being assessed in industry should be complete enough, have all the relevant functionality, be stable and usable enough to allow for proper assessment. First, even functionality that is not being directly evaluated can play a vital role in how industry practitioners see the candidate solution. An example is that of the workshop evaluation of the visualization aspects, as presented in P2. The artifact being assessed had the required functionality for this evaluation, but it was incomplete, i.e. it was not able to generate additional test cases than the ones used for the evaluation, in the time required. That meant that the domain specialist had a harder time assessing if the visualization was sufficient for the purpose. As soon as all the functionality was present, the detail of feedback that domain specialists could provide also increased, as they were better able to see the results of their interaction with the ISBST system in the solutions it proposed. This is especially true for search-based software testing,
which is a comparatively novel concept that domain specialists have a hard time envisioning. In later studies, candidate solutions were generated in real time, as a result of the interaction between the domain specialist and the ISBST system. The active interaction allowed industry practitioners to see the results of the interaction immediately. Thus, the concept was easier to explain, its usefulness easier to determine, and potential improvements easier to identify.

Second, there is a danger of causing decision fatigue in the domain specialists. Quick improvements in the tool may lead to several variants of the same system that differ only in terms of minor improvements. While regular feedback from industry practitioners is extremely useful, artifacts that are shown should exhibit clear improvements and integrate previous feedback. Thus, the limited time that industry practitioners have can be put to the best use, and their interest and engagement levels maintained.

Finally, any artifacts that are being assessed in industry should receive as much evaluation as possible in a controlled, laboratory setting. This is both to improve the artifacts themselves, to avoid the challenges presented above, and to develop and improve the assessment mechanisms. Issues that are not directly linked to the functionality being tested could also have a significant impact. For example, developing the ability to integrate, on the fly, new SUTs and new objectives required a certain amount of effort and was not essential to validating the ISBST system. However, during discussions with our industrial partner, the ability to showcase the capabilities of the ISBST system on a SUT proposed by the industrial participants on the spot, proved convincing and useful.

A final topic related to the technology transfer is that of integration with existing tools, artifacts, and processes. It is easy, when developing and transferring a type of technology to industry, to focus on the technology itself. The issue of using any resulting tools after the transfer process has ended, and implicitly how these tools interact with the existing context, is less often discussed. The Technology Transfer Model discusses the importance of training [48], first focused on industry champions and early adopters, then focused on enabling early adopters to train other interested users. In terms of search-based software testing, however, this appears to be harder to achieve. Search-based software tools have been developed and assessed, with open-source and industrial code [14, 15, 16]. Nevertheless, these studies were conducted by researchers, and ultimately do not discuss any tools that have been completely transferred, i.e. are being developed and used without the involvement of researchers. This suggests that the tools are still quite complex, and require specialized knowledge to maintain and use.
1.9.3 Communication and Visualization

One interesting problem that the ISBST system faced is that of communicating the results of the search to the domain specialists.

For individual test cases and candidate solutions, the ISBST system uses similar notations to those used in the target domain. This requires that information about any standards of visualization be available and collected early on. In the case of the ISBST system, standards already existed. The SUT focuses on input and output signals that can be visualized as line graphs. However, for different domains, different visualizations approaches may be relevant. This may be seen as a somewhat unimportant issue, as data visualization can always be developed to match. However, assessment of the ISBST system was greatly helped by the familiar visualization: domain specialists were better able to understand the solutions that the ISBST system was developing, to see the potential use of such an approach, and to provide constructive feedback on how the system could be improved.

The more problematic issue is that of visualizing the progress of the search. The large number of candidate solutions being generated and the high dimensionality of the search results make it quite difficult to display the progress of the search in a meaningful and helpful way. Von Lücken et al. [55] identify visualization of objective and trade-offs as non-trivial, even for two or three such objectives. Increasing the number of objectives adds to the difficulties. The work identifies three classes of visualization for many-objective problems. First, showing objectives in groups of two or three at a time, e.g. a scatterplot matrix. This approach shows all the information, but makes it difficult to visualize relations that exist in the entire objective set. The second class focuses on showing all objectives as bar charts, petal diagrams, or other representations. The third class consists of reducing the number of objectives, and displaying the resulting set. In spite of these alternatives, von Lücken et al. conclude that there is still a lack of intuitive and simple visualization techniques.

This means that, especially for domain specialists that are not experts on search-based approaches, the information could be difficult to understand and use. During the evaluation discussed in P7, practitioners often found it difficult to identify outlier candidate solutions due to information overload. The exact method by which the progress of the search, and the relative characteristics of the candidate solutions can be communicated to the domain specialists, is still to be developed. The experience of the ISBST system, however, shows that potential users of the system should be clearly informed about which candidate solutions exhibit extreme behaviors, with respect to each of the search objec-
tives. In addition, some means of assessing and presenting to the user how each of the scores for each search objective changes could allow them to better understand how the search is progressing. Undoubtedly, additional mechanisms can be developed that help domain specialists better understand the candidate population.

For the ISBST tool we developed and evaluated in this work, we used the scatterplot matrix as a way to visualize the candidate population. This allowed all the available information to be displayed. Domain specialists could then select subsets of objectives that they wanted displayed in more detail. Based on the feedback we received, the method was difficult to understand due to the large amount of information on the screen. Although the scatterplot matrix was provided with some tools to help with selecting candidate solutions and observing their performance across all the objectives, the approach was not intuitive.

Alternative approaches to scatterplot matrices have been proposed. Pryke et al. [56] propose a heatmap where columns are objective and parameter values, lines are individual candidates, and the color encodes the normalized value for each candidate. The work validates the heatmaps on problems with only two or three objectives, but the concept has been applied on many-objective problems as well [57].

While these visualization methods do show the available data, it might be difficult to identify “interesting” candidate for further development and use. Walker et al. [57] also propose a rank visualization. For multi and many objective algorithms, this relies on forming a partial ordering based on Pareto optimality, and using that partial ordering to make it clear to the decision maker which solutions are dominating and which are not. This technique could help domain specialists using the ISBST system in identifying solutions that are not dominated. This ordering, coupled with a clear indication to the user of the reason why certain solutions are in their respective positions, could help the ISBST system better communicate its findings.

Overall, more work in visualization of search information is required. Developing, integrating, and validating additional such visualization techniques could provide a better understanding and easier acceptance of search-based techniques, both in industry and in academia.

1.9.4 Assessment and validation

In addition to developing the ISBST system, this thesis also discusses some of the mechanisms used to evaluate and validate that system.
In academia, evaluation focuses on objective measures. We used experimentation to assess the ability of the ISBST system to generate solutions of increasing quality, as measured by the search objectives. The experiments also measured the diversity of the resulting solutions. For this thesis, a measurement of the degree to which the behavior space is explored has also been used. These are all measurements that focus on the capabilities of the ISBST system, but exclude the context. The experiment presented in P5 also uses the NASA-TLX assessment questionnaire to assess the mental demands, and level of fatigue, that the ISBST system places on its users. This is an important aspect to consider when the ultimate goal is technology transfer.

The same evaluation methods were less useful in industry. The number of industrial participants that were available to participate in evaluation was relatively small, and the context was quite specialized. As a result, assessment methods that focused on generalizability and a high volume of data would not be useful. Moreover, if the goal is technology transfer, it is more important to focus evaluation on how the tool or technique can be improved to match the context rather than to try to generalize findings across contexts. For the industry evaluation, the analysis of the search results was augmented by more qualitative techniques that focus on in-depth information: workshops, semi-structured interviews, and think aloud protocol.

It is important to note that assessment mechanisms appropriate to the context and the goal are essential for successful evaluation and later transfer of technology to industry. Concerns and priorities that industry practitioners and industrial partner express also form a useful starting point for potential evaluation methods.

1.9.5 Answers to the research questions

This section will provide a brief overview of the answers to the research questions.

- **Development: How can search-based software testing be developed for use in industry?** A direct answer to this question is that the current iteration of the ISBST system provides a model of how a search-based software testing system can be developed and adapted for use in industry. We show that interaction allows domain specialists to use their experience and knowledge to guide the SBST system. Design science research [52, 53] provides a useful framework for the development of such a system for use in industry: by using the requirements of industry, as
well as the methods and techniques validated in academia, as inputs to an iterative design process. It also encourages repeated field testing, to ensure the relevance of the proposed solution, and to allow feedback and additional requirements to be collected.

• **Deployment:** How can an SBST/ISBST tool be deployed to industry? The technology transfer model [48] provides a framework that the deployment can be based on. The repeated validations, in academia and in industry, allow researchers to develop their proposed tool and to ensure that it is suitable for the industrial context where it is deployed. In the current example, the ISBST tool’s flexibility was improved, to allow it to test new SUTs without the need to manually instrument code. The answer to this question is that deployment can be achieved as a multi-step process, with repeated validation of the proposed solution and with considerable tailoring to the industrial context and to the needs of the practitioners.

• **Evaluation:** How can an SBST/ISBST tool be evaluated and validated throughout the deployment steps? SBST systems have been evaluated in academia and industry, focusing on showing their usefulness and performance. When focusing on the deployment process, two types of evaluation are necessary. First, the evaluation of the tool being transferred, on how well it performs and how effectively is solves the engineering problem. This type of evaluation leads to improvements in the tool itself, and makes it better fit the need of the industrial practitioners. Different methods of evaluation are suitable for the academic and industrial contexts, and even where the same methods apply, their result might differ. The ISBST system performed better in findings injected bugs in a controlled, laboratory environment, than it did in an industrial context. Specific evaluation tools may be required for each of the stages of the deployment. For example, evaluating the interaction between domain specialists and the ISBST system, the communication and visualization of information, could only be conducted conclusively with domain specialists in an industrial setting. Evaluating the interaction at earlier stages provided useful information, but it was only the validation in industry that could provide a convincing assessment of its suitability. The second type of evaluation focuses on assessing the progress of the deployment process. This is harder to evaluate, and further work in necessary to determine how the progress of the deployment itself can be measured, and how the effort and time still needed to complete the deployment can be estimated.
1.10 Threats to Validity

The threats to the validity of the studies presented in this thesis should also be discussed.

First of all, all the studies were shaped by the industrial context that was considered for the transfer. For the industrial evaluations, discussed in P2, P4, and P7, this means that a small number of industrial practitioners participated. This does limit the generalizability of the findings. In response, the thesis makes no claims to generalizability: the goal was to develop and transfer an interactive search-based software testing system, as a pilot study. In addition, while the number of participants was limited, the information gathering methods chosen, i.e. workshops, semi-structured interviews, the think aloud protocol, allowed for detailed data collection and feedback.

Even for the studies that were not conducted in industry, the industrial context shaped the ISBST tool itself, the methods for assessment, and the goals of the evaluation. For example, our choice of focusing on interaction as a key mechanism to shield domain specialists from the intricacies of the search-based software testing system was driven by the industrial context. This choice may not be as valid in other contexts.

Previous sections discuss the issue of evaluation, and how mechanisms for evaluating the ISBST system had to be developed and tailored for the context and participants. This allowed us to ensure that the evaluation methods we used were flexible enough to be relevant for the study, and meaningful enough to allow us to draw conclusions regarding the ISBST tool. However, this tailoring also introduces certain threats to the validity of these studies. Every effort has been made to ensure that the assessment of the ISBST system is objective and complete, but there is a chance that relevant aspects may have been overlooked.

Lastly, the issue of communication has been discussed above. This includes both communicating to the user the characteristics of a particular test case and that of communicating the overall view of how the search is progressing, what types of behaviors have been investigated and which test cases show extreme behavior according to some search objective. This involves communicating large amounts of complex information to the domain specialist, and there is a possibility that this communication was misunderstood. This presents a threat to the validity of the work, since misunderstanding the communicated information could alter the feedback and evaluations that practitioners provide. The issue of communication has been a concern from the early efforts, as shown in P2, and has been assessed in follow up studies as well. These efforts have lead to an
improvement in the visualization and data communication aspects of the ISBST system, but further work is necessary to completely address this problem.

1.11 Future Work

This thesis presents the development and deployment of an interactive search-based software system in a particular industrial setting. The process is not complete, though, and this section will discuss ideas for future work.

**Dynamic Validation.** The first direction is to complete the dynamic validation of the ISBST system. While the ISBST system has been deployed and evaluated in industry, evaluation in an active project over a longer period of time is still necessary. From an academic perspective, this will provide a complete roadmap on how a search-based software testing technique can be developed and delivered to industry.

In addition, the transfer of the ISBST system from validation in academia to static validation in industry has revealed a number of improvements and changes that had to be made to the evaluation methods. There is no reason to believe that a dynamic validation would not result in further lessons learned, further improvements, and additional method development.

**Generalizability.** The initial development and evaluation of the ISBST system presented here is limited to the context provided by our industrial partner. A potential next step is to extend the interactive search-based approach to other companies, contexts, and domains. This would allow us to further assess the generalizability of the findings presented here, as well as to identify areas where domain-specific and context-specific tailoring is necessary.

Part of the process of widening the application of ISBST to industry would be the incorporation of new search algorithms and development of hybrid approaches. Paper P6 evaluates a number of exploration-focused algorithms and concludes that they show promise, either on their own, in situations where search objectives are unclear, or to augment existing approaches. Hybrid approaches, suitably validated and deployed in industry, are part of the future work plans.

**Assessment methodology.** The ISBST system has been developed in parallel with assessment and validation techniques. Future efforts could also focus on evaluating a developing SBST system for deployment in industry, assessing how current technology transfer projects are progressing, and better understanding which methods are more appropriate for the academic and industrial contexts. The experience of the ISBST system has shown that methods that fit the academic environment are not guaranteed to transfer well to industry. Fur-
Further work is needed before a conclusion can be reached regarding which methods are more appropriate in each context, and what factors affect the performance of each assessment method.

The thesis only touches upon the assessment of information visualization, communication, and interaction with domain specialists. Further efforts in this direction, particularly on assessing how this communication is proceeding, and how it can be improved in various circumstances, is necessary. Visualizing large amounts of complex data is a non-trivial problem, but one which should be addressed before successful transfers of search-based techniques to industry can be done with confidence. In addition to improvements to the interaction of the ISBST system, and other similar systems, with domain specialists, the issue of interaction with a company’s software development and testing processes should also be studied in more depth.

Ultimately, it may be possible to combine these efforts and develop a search-based “development and deployment toolkit”, containing all the algorithms, approaches, interaction methods, assessment methods, and all the experience of previous applications. This would allow researchers to use existing experience and information in the effort of further tailoring and assessing search-based techniques in industry, and provide a practical benchmark for evaluating novel algorithms and techniques.

**Visualization and interaction**

The ISBST tool relies on interaction with the domain specialist to guide the search, build a useful fitness function, and provide an overall assessment of the candidate solution quality. As the validations in industry have shown, interaction with the domain specialist, and communicating the progress of the search and the relative qualities of the candidate solutions require additional improvement.

A potential direction for future work is to develop and incorporate additional visualization mechanisms, and then assess them in industry and academia. In addition to displaying search information through heatmaps and ranked lists, an augmented ISBST system could track the progress of the search and communicate to the domain specialist which search objectives show the greatest improvement, and which candidates are closer to the extreme observed values for those search objectives. Clear and meaningful communication with the domain specialists is a key component in any interactive SBST system. So further work to improve communication would be helpful in improving and deploying search-based techniques in industry.
1.12 Conclusions

This thesis presents the development and deployment of an implementation of Search-Based Software Testing (SBST) to industry, to support the testing process in an industrial context. Given the characteristics of the industrial context, the solution chosen to achieve this included an interaction component, aimed at enabling domain specialists to contribute their knowledge and experience without having to deal with the details of search-based software testing techniques.

The thesis presents the development, evaluation, evolution, and deployment of the ISBST system, from concept to static validation in industry. It also describes the development and use of evaluation mechanisms, and presents a road map for how technology transfer projects can be achieved for search-based software testing systems, using the Gorschek et al. [48] Technology Transfer Model as a framework.

The main lessons of the work presented in this thesis relate to the solution itself, the deployment of SBST tools in industry, and the evaluation of such tools and technology transfer projects. The solution has to be developed in close collaboration with the industrial partner, tailored to their needs and goals, and adapted to function with their tools and in their particular context. Such an approach ensures that the solution is relevant for the problem identified in industry, and ensures the commitment of the industrial partner to the success of the project.

The success of a technology transfer project also depends on the approach to deployment. First a tool has to be developed with the prospective users in mind: focusing on what kind of information they are likely to have or need, and how they will find it best to interact with the tool. This requires frequent and detailed evaluations, with industry practitioners providing the feedback. This also requires that the tool is developed with flexibility, usability, and robustness in mind. Flexibility is essential to incorporating feedback quickly, while robustness and usability ensure that the industry practitioners that evaluate the tool can form a good impression of the capabilities and potential of the tool and are able to provide useful feedback for its further development.

A final conclusion relates to evaluation approach. The frequent changes, improvements, and updates to the ISBST system point to the importance of continuous evaluation of the proposed solution. Evaluations in academia, and later in industry, allow researchers to correct problems as they become known, validate their assumptions, and build trust about the solution they are proposing. The evaluations should also include domain specialists, prospective users, and other stakeholders. This would ensure accurate, reliable, and meaningful
feedback, in addition to encouraging commitment and allowing practitioners to contribute their own ideas and suggestions to the project.

This thesis has presented the conception, development, and evaluation of an SBST system for deployment to industry. The system relies on interaction to enable domain specialists to provide their knowledge and experience without requiring specialized knowledge of search-based systems. Long-term, we hope that this work will provide a useful template for the transfer of search-based technology to industry, allowing a wider use of SBST in industry and providing additional information and a higher level of understanding of SBST, its benefits and drawbacks, and of potential improvements.
Chapter 2

A Concept for an ISBST System

Bogdan Marculescu, Robert Feldt, Richard Torkar

Abstract

Software is an increasingly important part of various products, although not always the dominant component. For these software-intensive systems it is common that the software is developed and assembled by domain specialists rather than by software engineers. To leverage the domain specialists’ knowledge while maintaining quality we need testing tools that require only limited knowledge of software testing.

Since each domain has unique quality criteria and trade-offs and there is large variation in both software modeling and implementation syntax as well as semantics it is not easy to envisage general software engineering support for testing tasks. Particularly not since such support must allow interaction between the domain specialists and the testing system for iterative development.

In this paper we argue that search-based software testing can provide this type of general and interactive testing support and describe proof of concept system to support this argument. The system separates the software engineering concerns from the domain concerns and allows domain specialists to interact with the system in order to select the quality criteria that will be used to determine the fitness of potential solutions. We describe empirical, industrial evi-
dence from designing an interactive search-based software testing system using the methodology.

2.1 Introduction

There is an increasing integration of software into many products. This makes software quality a relevant factor in the overall quality of all such products. However, systems engineers and integrators are not software engineering experts and, in particular, have little or no software testing experience.

One option to ensure proper quality of the software being integrated is to involve dedicated software engineers to handle software development and testing. There are, however, drawbacks to this approach. First, this approach is quite costly. This is all the more valid for smaller companies that do not have the resources to accommodate this expense. Another drawback is that software engineers do not have the domain expertise required to test a software component for the environment in which it will have to operate. Thus, if not adapted to the context, the software in question may have a lower quality as a system component, in spite of having high quality as a software component.

An alternative option is to package general testing solutions to be usable by non-experts in software engineering, i.e. systems engineers and integrators. This would allow the domain expertise of the systems engineers to be fully used, whilst still applying proven solutions for software testing.

Search-Based Software Testing (SBST) is an excellent fit for the latter option. It has been shown to be a good approach for many different types of testing [8, 9]. It consists of a very generic search component, while those components that are domain-specific are those that systems engineers and domain specialists have their expertise in. These domain-specific components are the representation of the problem and the software as well as the quality criteria used for evaluation.

The contribution of this paper, therefore, is to propose a search-based software testing system that will allow domain-specialist users to create test cases for the software they produce, without the need for specialized knowledge of software testing or search-based techniques.

2.2 Background

Search-Based Software Testing (SBST) is the application of search techniques to the problem of software testing. SBST has been applied to a variety of testing
problems [8, 9], from object-oriented containers [24] to dynamic programming languages [25] and invariants [58]. SBST is a part of the wider area of Search-Based Software Engineering (SBSE), a term coined by Harman and Jones in 2001 [7], but in use earlier, see e.g. [11, 13, 22]. Since search-based approaches have been applied across the software development life-cycle [26], it is reasonable to expect that any conclusions referring to the general area of SBSE can be applied to the particular case of SBST.

The context of the system proposed in this work is that of software testing performed by domain specialists with relatively little knowledge of software testing or search-based techniques. The domain specialists would have extensive knowledge of the capabilities of the system-under-development, its own context and the limitations placed upon it, as well as the quality foci that they would have to pursue for each component of that system.

The combination of non-specialist software developers and the importance of domain knowledge and limitations makes it impractical to develop a fitness function up-front. To develop an appropriate fitness function for the component under test, the domain specialist would have to interact with the system and make adjustments to the criteria being used.

2.3 Search-Based Testing System

2.3.1 Running Example

To better illustrate the concepts discussed in this section, we will present an anonymized industrial example. The application we will use is that of a controller enabling a joystick or set of joysticks to handle a mechanical arm. The inputs for the controller software are the joystick signals and sensors that indicate the speed of the basket at the end of the crane. The outputs are the signals to the hydraulic pumps that drive the arm.

The System Under Test (SUT), in this case, is the software for the controller component. The goal of the Search-Based Software Testing System is to generate test cases that ensure the system’s compliance to quality standards, ensure that no constraints are broken and discover any additional faults or unexpected behavior.

The Search-Based Software Testing System is the result of applying the methodology presented in this work in the company in question. The system is meant to be tailored for the specific context and company it is expected to
function in, yet be general enough to enable domain specialists to test new applications within the confines of that context.

2.3.2 Overview and Components

The figure 2.1 shows the structure of a complex Interactive Search-Based Software Testing system developed using the proposed methodology.

![Diagram of ISBST System](image)

**Figure 2.1: Overview of an ISBST system.**

**Outer Cycle.** The outer cycle is an interactive search-based system that uses the human domain specialist as a fitness function. It mediates the interaction between the domain specialist and the system by means of a component called Interaction Handler. For the purpose of this discussion we call a potential solution or a solution candidate, any individual that is part of the population the human user is expected to evaluate.

The purpose of the Interaction Handler is to display the potential solutions shown to the human domain specialist and to collect their feedback. Feedback, in the type of system being proposed can refer to three separate issues:

- **Solution Candidate Feedback.** This describes feedback related to the solution candidates. In addition to selecting potential solutions for the next generation, the human domain specialist may assign values to each solution candidate they select for the next generation, giving them an evolutionary advantage.

- **Display Feedback.** This describes feedback related to the way solution candidates are displayed, the number of candidates displayed, and any additional information that is available or can be made available. Considering our running example discussed above, a domain specialist may
choose to see memory required, response times for the output signals or discrepancies between expected output signals and actual output signals, in addition to the pass or fail status of each suite.

- **Quality Focus Feedback.** Since a search-based system can generate more solution candidates that a human can be expected to evaluate, some internal mechanism exists to enable a preliminary selection of potential solutions. This type of feedback allows the domain specialist to set or change the criteria by which this preliminary selection is performed. As the search for appropriate test cases goes on, it may become necessary to adjust the quality foci that set the selection criteria. As an example, an initial requirement of the joystick controller in our example may concern appropriate timing or accuracy of the output signal. Once the module is considered satisfactory from that perspective, searching for large variations or undesired behaviors may become more important. This type of feedback would allow the domain specialist to alter the focus of the search without restarting the search, and thus preserving the characteristics of the solution candidates already in the population.

The replacement of the fitness function with a human domain specialist restricts the number of potential solutions that the system can process in this manner. The additional information that the human can provide is an attempt to compensate for the lower number of solution candidates being processed by improving the selection mechanisms internal to the system.

**Inner Cycle.** The inner cycle is a search-based software testing system that uses a flexible fitness function. The purpose of this system is to generate and select the best solution candidates for the human domain expert to evaluate. This approach allows a system to explore a wider solution space, while still allowing on the human domain expert to apply their experience and insight.

The inner cycle itself has two components:

- **Search Component.** The purpose of the Search Component is to encapsulate the algorithm that creates the new generation of potential solutions. Encapsulating this component allow the existing algorithm to be changed, should the need for such a change arise.

- **Intermediate Fitness Function.** This component serves the purpose of the fitness function in any search-based system: it assigns each potential solution a fitness value. The difference consists of allowing changes to be
made to this component during the search process. Such changes originate in the feedback the human domain specialist provides and allows them to influence the direction of the automated search as well as performing their own selection.

The purpose of this component is not to replace human input, but rather to provide an initial screening of solution candidates, so that only those solution that are most likely to be successful are analyzed by the human domain specialist.

The interaction between the inner and outer cycles will be achieved through the populations of candidates, the evaluations made by the domain expert and the feedback that will guide the Intermediate Fitness Function. The generation and selection of the population, as well as the internal workings of the inner cycle will be hidden from the domain expert.

2.4 Validation and Discussion

Validation efforts are focused on the development of a proof of concept system. This system will be developed in cooperation with an industrial partner and the initial validation will take place in that context.

The system presented here is designed specifically to address situations where domain expertise is the deciding factor in successful testing. This can be due to the complexity of the system under test and the influence external factors may have in its operation, as well as limitations in terms of the resources available for testing.

2.5 Conclusions

This paper has proposed a search-based software testing system designed to allow domain specialists with little software testing expertise to develop test cases for their applications. The value of such systems would be especially relevant for contexts were software testing experts are not available or where domain knowledge is the deciding factor in the success of the testing process.
Chapter 3

Objective Re-Weighting to Guide an Interactive Search Based Software Testing System

Bogdan Marculescu, Robert Feldt, Richard Torkar

Abstract

Even hardware-focused industries today develop products where software is both a large and important component. Engineers tasked with developing and integrating these products do not always have a software engineering background. To ensure quality, tools are needed that automate and support software testing while allowing these domain specialists to leverage their knowledge and experience.

Search-based testing could be a key aspect in creating an automated tool for supporting testing activities. However, domain specific quality criteria and trade-offs make it difficult to develop a general fitness function \textit{a priori}, so interaction between domain specialists and such a tool would be critical to its success.
Objective Re-Weighting to Guide an Interactive Search Based Software Testing System

In this paper we present a system for interactive search-based software testing and investigate a way for domain specialists to guide the search by dynamically re-weighting quality goals.

Our empirical investigation shows that objective re-weighting can help a human domain specialist interactively guide the search, without requiring specialized knowledge of the system and without sacrificing population diversity.

3.1 Introduction

Software is often developed as one part of a more complex system, by companies whose core competencies lie in other fields. Such companies often lack the software development and testing expertise needed to perform extensive software quality assurance, choosing instead to focus their efforts on maintaining and developing vital domain-specific knowledge. This reflects the reality that the quality of software-intensive products depends on a series of trade-offs, and software quality is just one concern among many. Increasing a company's focus to include software engineering and testing would incur significant costs, without guaranteeing a significant increase in the overall level of quality of the product in its entirety. An alternative to this approach is to use a pre-packaged software testing toolbox to enable domain specialists to focus on applying their experience and knowledge of the domain, rather than developing software testing skills. In this context we use "domain specialists" to denote systems engineers and other specialists in their own fields that have to develop, use and test software. The importance of knowing domain-specific constraints and trade-offs outweighs that of achieving proficiency in software testing.

It is in this context that interactivity becomes important. A pre-packaged software testing toolkit is difficult to develop and optimize before the specifics of the application become known. Moreover, the precise focal points of the testing process may change from one project to another or may vary in time within the same project, further emphasizing the importance of integrating domain knowledge into any effective testing tool.

This paper focuses on the problems met by domain specialists in testing software and proposes a solution to address, or at least alleviate, these problems.

In section 3.2, we consider existing approaches to interactive evolutionary search, and discuss how our approach differs from them. Section 3.3 describes the industrial context and a system we use as a running example. Then, section 3.4 describes in more detail the objectives of this study and the way the
system was designed to meet those objectives. A closer look at the implementation of that design can be found in section 3.5.

A brief empirical evaluation is described in section 3.6, with some of the threats to the validity of the study discussed as well. Section 3.7 concludes.

3.2 Related Work

Search Based Software Engineering (SBSE) is a term coined by Harman and Jones in 2001 [7] to describe the application of metaheuristic search techniques to software engineering problems, e.g. [11, 12, 13]. The branch of SBSE concerned with testing problems is known as Search Based Software Testing (SBST) and has been applied to several types of testing problems [8, 9], from object-oriented containers [24] to dynamic programming languages [25]. However, there has been very few studies considering interactive SBSE.

Feldt [27] described an interactive development environment where tests are created as the engineer write the program code or refine the specification. The system used the interactions of the engineer to help guide the search but the effect on the fitness function was indirect. Feldt [12] and Parmee et al [59] considered the use of interactive search to explore engineering designs and better understand design constraints.

In a previous paper [60], we proposed a system that combines several of these concepts, e.g. interacting with the system once in a number of generations, rather than each generation [61]; and adds that of interacting with a ISBST system by means of allowing the human to modify the fitness function. The Intermediate Fitness Function (IFF) describes the current goal and contains all the information available at a given moment. As a result, it is reasonable to assume that it should be updated as that understanding changes or becomes more refined, or as new information becomes available.

3.3 Industrial Context

The approach presented in this paper was developed as a result of input from our industrial partner and is shaped by the context where they operate.

Our industrial partner develops products that involve embedded software, but where software is not the main consideration. As a result, knowledge of the domain and domain specific trade-offs has a greater impact on the overall quality of the product than expertise in software development and testing. This type of
situation emphasizes the importance of extensive knowledge of the domain and experience with the context and limitations of the application being developed. These allow a domain specialist to better assess quality characteristics that the complete product must have, as opposed to the quality level of the software as a separate entity.

We will use a running example to better illustrate the challenges we faced and to clarify the approaches we used to address these challenges. The example is that of a control mechanism for a electric motor, e.g. powering a mechanical arm. Due to limitations of the motor itself and the potential for damage in what the mechanical arm is handling, an average filter is necessary to convert sharp changes in input into a smoother signal for controlling the motor. The example is based on a model filter provided by our industrial partner. The filter is relatively simple, but common enough to be included in the standard toolkit of commonly used components.

The system under test (SUT) is the aforementioned average filter. In this particular example, there are three major quality goals to be accomplished. First, since the filter’s purpose is to smooth a signal for use as input in a motor, it is important that the output is free from discontinuities which might damage the motor itself, the arm or anything the arm might be handling at the time. Second, there are limits to what inputs are acceptable to the motor, for the same reasons. Last, it is important that an input signal is as short as possible, to enable any test case to be human readable and understandable and to allow assessment of the test case in a reasonable amount of time.

During the project we held discussions with our industrial partner and their clients and took part in one of the training sessions for developing software using a domain-specific tool.

The approach of selecting domain specialists and giving them training in developing software using a domain specific tool illustrates the relative importance of domain knowledge and experience compared to expertise in software development and software testing. This guided our approach toward developing a support tool for domain specialists rather than trying to capture domain-specific knowledge for the benefit of software engineers. In addition to domain knowledge and experience, the system must account for the fact that the domain specialists may come from different, albeit related, backgrounds and are developing different and quite disparate products.

As a result of these efforts, we have developed an Interactive Search-Based Software Testing (ISBST) system. This system searches for test cases that break, or come close to breaking, the quality goals stated above, under the guidance of a domain specialist.
3.4 Objective and Method

As touched upon in the previous sections, the software in this context is developed by domain specialists rather than software engineers. The main problem identified as a result of our discussions with our industrial partner was that of creating test cases that cover a significant part of the input space.

Currently, test cases are created ad-hoc for each module, based on the developer’s intuition and experience with similar modules in the past. While this allows past lessons to be incorporated into the development work, it also means that new problems are hard to identify.

To address this problem we propose an ISBST system that would benefit from the exploratory potential of search based techniques, while still benefiting from the experience and intuition that domain specialists rely on.

This study aims to answer the following research question:

RQ1: How can interaction between a human domain specialist and a search based testing system be achieved?

To answer this question we propose Interactive objective re-weighting as a means of interaction, especially in situations where:

- The domain specialist is not a software engineer and cannot be required to obtain expertise in software engineering.
- The domain specialist cannot be expected to evaluate a significant proportion of the candidates, due to the large number of candidate solutions as well as their complexity.
- Maintaining population diversity throughout the process is an essential part of the approach.

A secondary question is:

RQ2: If such an interaction can be successfully achieved, what limitations or guidelines can be identified to ensure that population diversity is not negatively affected?

To answer these questions, we have used a number of interviews and workshops, conducted with our industrial partner, as a basis to develop an SBST system that would enable the domain specialist to interact by Interactive Objective Re-Weighting (IORW).

The Interactive Objective Re-Weighting approach relies on the notion of “Interactive Fitness Function” or IFF. The IFF is a fitness function that can be dynamically modified during the course of the search, to better match the
current understanding of the requirements on the system. The IORW is one means of achieving a workable IFF.

The IFF for a candidate \( j \) is computed as follows:

\[
IFF(j) = \sum_{i=1}^{n_{\text{Objectives}}} \text{Weight}_i \times \text{Value}_{i,j}
\]  

(3.1)

where \( IFF(j) \) is the fitness value of candidate \( j \), \( \text{Weight}_i \) is the current weight of the objective \( i \), and \( \text{Value}_{i,j} \) is the fitness value of candidate \( j \) measured by objective \( i \). An objective \( k \) can be deselected from the computation by having \( \text{Weight}_k = 0 \).

Our approach is to use the IFF as a surrogate, evaluating a number of \( n \) optimization steps, between interaction events. An ‘interaction event’ is one interaction between the system and the domain specialist. It consists of the system displaying the current weighting and the best candidates according to that weighting, and of the domain specialist conducting a re-weighting if they feel it is needed.

This will help reduce the burden on the domain specialist without sacrificing population diversity and size. The exact value of \( n \) may vary from one system to another.

The IORW approach is presented in more detail in Algorithm 1.

The default setting for the IFF function is that of having the weights for all objectives as having an equal weight, i.e. \( \forall j, \text{Weight}_j = 1 \).

Thus, the domain specialists adjust the weights of the various objectives, based on their understanding of their relative importance and, in doing so, shape the IFF for the next set of optimization steps. The objectives are defined and presented in domain-specific ways, thus allowing domain specialists to dynamically generate a new IFF using only concepts that they are already familiar with and actively using. This can be done without requiring the domain specialist to achieve proficiency in software engineering in general, and in SBST in particular.

### 3.4.1 System Design

Figure 3.1 shows the design of the ISBST system. The system is split into two major components. The inner cycle deals with generating the candidates, running the candidates as inputs to the SUT and evaluating their fitness according to the IFF. The outer cycle handles the interaction with the domain specialist, i.e. displaying candidates and interpreting the feedback provided.
set the IFF to default;
set currentStep = 0;
while acceptableSolution == false do
    if currentStep == interactionStep then
        begin Domain Specialist Interaction Step;
        solutions are displayed and evaluated by the domain specialist;
        if domain specialist accepts one of the proposed solutions then
            acceptableSolution = true;
        else
            acceptableSolution = false;
        end
        if domain specialist adjusts objective weighting to better reflect
        their goals then
            change the IFF to reflect the objective re-weighting;
        else
            IFF remains unchanged;
        end
    else
        perform optimization step with the current IFF;
        currentStep += 1;
    end
end

Algorithm 1: Dynamic modification of the IFF via IORW
Objective Re-Weighting to Guide an Interactive Search Based Software Testing System

The design is shaped by the need to separate those components of the system that require interaction from the domain specialist, from those that can be fully automated. By achieving this separation, the exposure of the domain specialist to the underlying SBST system can be better controlled.

The system keeps the overall structure presented in [60], while making allowances for the increasing complexity inevitable in practical implementation.

The Outer Cycle. The outer cycle is an interactive search based software system, where the domain specialist is presented with a set of candidates and, based on those candidates, they decide if the current objective weighting is appropriate. If it is not, then the objective weighting is changed to reflect the current understanding of the current quality needs.

The design uses some of the fatigue reduction concepts defined in [61]: e.g. requiring human interaction every $n$ of generations rather than every generation, reducing the overall number of solutions that have to be evaluated by the human engineer, and focusing on aspects that do not require detailed evaluation of each individual solution. The outer cycle captures those interactions, while abstracting away the automated evaluations and removing the need for direct involvement with the technical details.
3.5 Implementation

The human domain specialist guides the automated evaluation by providing the objective weighting that serve as input for dynamically generating the IFF. Moreover, this is done after the domain specialist is shown the fittest candidates, as evaluated by the current IFF. All this functionality is provided by the components of the outer cycle.

**The Inner Cycle.** The inner cycle contains the search based software testing system developing the potential solution candidates, dynamically generates the IFF from the objective weights, performs the automated evaluation of candidates, and interfaces with the outer cycle and the SUT.

The interface to the SUT is concerned with the modules required to run the SUT for evaluation. Apart from information strictly necessary for interaction, the inner workings of the SUT are hidden.

The inner cycle functions as a traditional SBST system, with the **Searcher** generating the candidates, using the SUT to run those candidates, and evaluating them by means of the IFF.

3.5 Implementation

The outer cycle uses a combination of CoffeeScript, a variation of JavaScript aimed at simplifying development, and the Data-Driven Document (D3) library to display candidates and obtain feedback from the domain specialist. D3 is a JavaScript library for the manipulation and display of data in a browser. It allows visualizations to be dynamically created, without acting to change the data itself. Since the outer cycle is concerned with displaying the candidates, the combination of Coffescript and the D3 library is a good fit for the purpose.

The inner cycle and the SUT Interface are developed in Ruby. This was chosen for the relative ease with which it interacts with other applications, e.g. other SUTs, interfaces to simulators or interfaces to hardware test benches, enabling the ISBST to be more extensible overall.

First, a population of candidate solutions is generated by the **Searcher**. Each candidate is converted to a candidate object and run through the SUT or, by means of the SUT Interface, through the hardware test bench. Each candidate then receives a fitness score for each of the quality objectives that were selected. These scores, together with the respective weights, combine to form the IFF score, as discussed above.
Encoding the Candidate Solutions. Our example is that of an average filter, that receives a signal describing the desired level of output and a number of steps to reach that level. In practice, our industrial partner uses a discrete encoding for each signal. Our system uses a Frame object to contain the input and output values at a given discrete step. Both are real numbers, with the outputs being computed by running the inputs through the SUT.

The candidate object consists of the ordered set of Frames, and also encapsulates the fitness values measured for each of the objectives.

When user interaction is required, the candidate objects are packaged into a JavaScript Object Notation (JSON) file and made available to the outer cycle. Search-related meta-information, e.g. fitness values for each objective, is included in the candidate object, and therefore available in the outer cycle. This enables the system to more easily adapt to changes in display requirements.

3.6 Evaluation

This section presents an initial evaluation of the ISBST system presented above, and in particular of the mechanism of Interactive Objective Re-Weighting (IORW) used to handle the interaction with the domain specialist.

The criteria by which the success of this evaluation will be judged are as follows:

- A system can be developed that enables a domain specialist not familiar with SBST to guide a search using IORW. Successful use of IORW would prove the viability of the ISBST system we are proposing.
- The interaction between the domain specialist and the ISBST system can be shown to have an effect. In addition, identifying any potential differences between interaction strategies, would enable further studies to refine our understanding of the interaction, its benefits and limitations.

The evaluation uses the SUT described above, which is a Ruby implementation of the filter described in section 3.3. The filter is a component of the standard library used by our industrial partner and offered to their clients. It was chosen for being a very simple, but common, component that is actively used in practice. For the purposes of this evaluation, a number of discontinuities were injected, to simulate faults in the software.

- Discontinuity (D). Measures the number of discontinuities found in a candidate. The aim is to maximize this objective.
3.6 Evaluation

- Upper Limit (UL). Given an upper limit for the input signal, this objective measures how close a given value is to that limit and, if the limit is exceeded, the difference between the limit and the actual value. The objective is to be maximized, as the goal is to find the signals that might exceed the maximum value.

- Length (L). For the purpose of this evaluation, we consider that a candidate should have close to a given length. As a result, candidates with lengths lower than \( \text{Min} \) frames or higher than \( \text{Max} \) receive a positive cost, while candidates with a length between the two values receive a negative cost. This objective is to be minimized.

By formulating the interaction in terms of objectives and their weights, the domain specialist interacts with representations and objectives that are familiar, and do not have to acquire additional software engineering skills.

An instance of IORW signifies that the domain specialist has re-evaluated their goals and priorities, either in light of new information becoming available from outside the system, e.g., changes in goals, domain specific constraints, or as a result of candidates noticed in the population of candidates being displayed.

In our example, a domain specialist starts their analysis of the SUT and their first priority is identifying discontinuities in the system. The second most important objective is the length of the candidate.

A set of interaction events were performed at regular intervals, once every 250 optimization steps. The weights are given as integers, from 1 to 5.

The set of interaction events in Table 3.1 shows an example of how a domain specialist can interact with the system, by specifying the relative importance of the respective objectives. In the example there, the first run is the default setting, with all objectives of equal weight. The domain specialist can choose to focus on one objective, e.g., Event 2, or try to find a more balanced approach, e.g., Event 3.

Through all the interaction events, the domain specialist evaluates the importance of each objective and the candidates they are shown, with the minutiae of the underly process being hidden from view.

Table 3.2 shows the mean fitness values of the displayed candidates for each quality objective.

An interactive objective re-weighting event, as expected, results in changes in the candidates selected for display to the domain specialist, as well as in changes to the IFF and the candidates that are generated. This increases the confidence that the re-weighting has a significant impact on the ranking of the
Table 3.1: Event sequence for investigating the impact of a single interaction event on the candidates. The abbreviations: Ev - Event, D - Discontinuity, UL - Upper Limit, L - Length

<table>
<thead>
<tr>
<th>Ev</th>
<th>Description</th>
<th>Weights</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>D</td>
</tr>
<tr>
<td>1</td>
<td>Describes the status at the beginning of the first interaction event. Initially, each objective is assigned an equal weight</td>
<td>1</td>
</tr>
<tr>
<td>2</td>
<td>The weight of the Discontinuity objective is raised to 5, signifying that the Discontinuity objective is about 5 times more important than the others</td>
<td>5</td>
</tr>
<tr>
<td>3</td>
<td>The weight of the Length objective is raised to 3. Finding a shorter candidate is more important than the upper limit, but less important than finding a discontinuity.</td>
<td>5</td>
</tr>
<tr>
<td>4</td>
<td>The relative importance of Length and Discontinuity are reversed. Finding a discontinuity is more important than the upper limit, but less important than finding a short candidate.</td>
<td>3</td>
</tr>
<tr>
<td>5</td>
<td>Return to the prioritization in step 3.</td>
<td>5</td>
</tr>
</tbody>
</table>

Table 3.2: Means objective values of displayed candidates

<table>
<thead>
<tr>
<th>Event</th>
<th>Discontinuity</th>
<th>Upper Limit</th>
<th>Length</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>2.333</td>
<td>13755</td>
<td>-6.2</td>
</tr>
<tr>
<td>2</td>
<td>3.6</td>
<td>12634</td>
<td>-0.333</td>
</tr>
<tr>
<td>3</td>
<td>3.8</td>
<td>5805</td>
<td>-0.533</td>
</tr>
<tr>
<td>4</td>
<td>2</td>
<td>3570</td>
<td>-8.866</td>
</tr>
<tr>
<td>5</td>
<td>3.333</td>
<td>15032</td>
<td>-2.266</td>
</tr>
</tbody>
</table>
candidates, as they are displayed to the domain specialist. Since the ranking is based on the IFF, it is reasonable to claim that the re-weighting has an impact on that function and, through it, on the SBST system.

The issue of validity also needs to be discussed. The study presented here resulted from our efforts with one company, in one particular domain. To alleviate this problem a general SUT was selected and, by treating the SUT as a black box, the ISBST system is not tied to any particular implementation. These considerations indicate that the approach could be generalizable. To ensure such generalizability, however, further efforts are needed.

An additional issue is that the SUT used for the evaluation is a simple component. While the filter itself is simple, it cannot be considered quite a toy example: it is widely used and present in a toolbox of common components.

3.7 Conclusions

In this paper we presented an Interactive Search-Based Software Testing system where the main method of interaction is that of objective re-weighting. The approach is a novel type of interaction based on our experiences with our industrial partner and their clients, and is based on their current testing practices and concepts. The approach does provide the flexibility and extensibility needed and allows domain specialists to interact with an ISBST system. The empirical evaluation indicates that the approach is viable; objective re-weighting can help a human domain specialist, unfamiliar with SBST or SE, interactively guide the search by adjusting their objectives alone.
Objective Re-Weighting to Guide an Interactive Search Based Software Testing System
Chapter 4

Practitioner-Oriented Visualization in an Interactive Search-Based Software Test Creation Tool

Bogdan Marculescu, Robert Feldt, Richard Torkar

Abstract

Search-based software testing uses meta-heuristic search techniques to automate or partially automate testing tasks, such as test case generation or test data generation. It uses a fitness function to encode the quality characteristics that are relevant, for a given problem, and guides the search to acceptable solutions in a potentially vast search space.

From an industrial perspective, this opens up the possibility of generating and evaluating lots of test cases without raising costs to unacceptable levels. First, however, the applicability of search-based software engineering in an industrial setting must be evaluated.

In practice, it is difficult to develop a priori a fitness function that covers all practical aspects of a problem. Interaction with human experts offers access
to experience that is otherwise unavailable and allows the creation of a more informed and accurate fitness function.

Moreover, our industrial partner has already expressed a view that the knowledge and experience of domain specialists are more important to the overall quality of the systems they develop than software engineering expertise.

In this paper we describe our application of Interactive Search Based Software Testing (ISBST) in an industrial setting. We used SBST to search for test cases for an industrial software module and based, in part, on interaction with a human domain specialist. Our evaluation showed that such an approach is feasible, though it also identified potential difficulties relating to the interaction between the domain specialist and the system.

4.1 Introduction

Software is often developed as only one component among many in complex, engineered systems. In such a situation, not all system developers can be expected to have software engineering expertise. Nevertheless, their domain knowledge is often critical in creating and selecting test cases. Search-based software testing can automatically create software test cases and thus potentially tap into a broader experience base by presenting these test cases to the developers, and allowing them to interactively select the most meaningful ones [8, 9, 27]. The human experts, henceforth referred to as “domain specialists” are, therefore, an integral part of software development, using their knowledge and experience to make those trade-offs.

The tests thus developed need to be represented in a way that system developers, domain specialists, and even users, can understand and that enables an informed selection to be made. This can be achieved by matching test representations to the domain, rather than to the traditional programming and testing languages. Adopting domain specific representations, thus, allows information to be presented in ways that are already familiar and avoids the added burden of adapting to new representations.

In this paper we investigate the effectiveness of such representations for interactive, semi-automated testing of software for embedded control systems developed by a Swedish systems engineering and manufacturing company. The company’s input was crucial in developing a prototype ISBST system, evaluating our assumptions, and learning from these efforts. Our contribution is one of the first deployments and evaluations of an Interactive Search Based Software Testing (ISBST) system in actual, industrial practice.
In Section 4.2 we present existing efforts with respect to interactive evolutionary search and related approaches, and discuss how our approach differs from them. Section 4.3 describes the industrial context that forms the basis of this work. Our approach is described in Section 4.4, together with a brief evaluation and a discussion on the lessons learned from this effort in Section 4.5. Section 4.7 concludes the paper.

4.2 Related Work

A term coined by Harman and Jones in 2001 [7], search based software engineering (SBSE) is the application of metaheuristic search techniques to software engineering problems, e.g. [11, 12, 13]. Search based software testing (SBST) is a branch of SBSE that deals with testing problems and has successfully been applied to several types of testing problems [8, 9], from object-oriented containers [24] to dynamic programming languages [25].

An important concept for search-based systems is that of fitness function. The fitness function can be seen as “the characterization of what is considered to be a good solution” [7]. The fitness function is used to select the best solutions in a population, and to guide the search towards good solutions.

Takagi defines Interactive Evolutionary Computation as “an EC that optimizes systems based on subjective human evaluation” [10]. This approach relies on human interaction to evaluate the solutions being developed by the interactive search system, allowing for situations where the choice of solution is dependent on “human preference, intuition, emotion and psychological aspects” [10]. The original paper refers to art and animation, graphics and image processing; in general applications where the evaluation of a candidate has a strong subjective component. A prominent example of this is Picbreeder [62], an online service where users evolve images, in a collaborative setting, using interactive evolution. The users’ aesthetic preferences drive the evolution, not any objective goal.

Nevertheless, we think that this can be generalized to any type of candidate evaluation where not all the necessary information, e.g. domain specific knowledge, experience, background information, or intuition, can be modeled into the system or encoded in an automatic evaluation approach [12, 60]. From the perspective of the system, implicit knowledge can be seen as subjective evaluation, thus avoiding the need to duplicate existing expertise.

Tagaki also identifies the problem of human fatigue [10]. This is a problem that arises when a human user has to perform a large number of interactions
with the system. This is an issue for any interactive system, since a human suffering from fatigue will not provide the level of analysis and decision making necessary to perform their duties appropriately. Therefore, a key element in an interactive system will not function properly and may even hinder the system’s ability to evolve a good solution.

Search based approaches have already been used as exploratory tools, in situations where there is an incomplete knowledge of the search space. Feldt [12] describes genetic programming being used to explore potential designs for aircraft arresting system software, and identifies the importance of obtaining problem-specific knowledge early in the design process. Parmee et al. [59] introduces an Interactive Evolutionary Design System to support the early stages of design. It identifies the importance of capturing “design knowledge through extensive designer interaction”.

SBST systems, like EvoSuite [63], require software engineering expertise to use. Adding interactivity to such a system in our context would require the domain specialists to acquire additional skills in software engineering, a process that would be costly in terms of time and resources.

Our system differs from previous approaches precisely in the issue of using the current domain specific representation as a base for interaction, while trying to shield the domain specialist from the software engineering specific details.

4.3 Industrial Context

Our industrial partner develops hydraulic and electronic products for off-highway vehicles and machinery. These products use embedded software for a variety of applications, from steering and transmission systems to sensors and displays, where software is an important but not the main component.

In addition, the company provide their customers with a software development tool, specifically designed to allow domain specialists to modify and develop their own embedded software, to be used with general purpose controllers. While the exact applications may vary greatly, in all these cases domain expertise outweighs in important software engineering expertise.

As a result, while software is an important component in their respective products, they focus their efforts and resources on other components.

In such a context, the quality of the resulting system depends on that of the software, but not exclusively so. Trade-offs may be needed, e.g. restrictions on software capabilities due to the need to use more robust and less capable
hardware, that cannot be known ahead of time or may emerge during the design process.

A domain specialist has knowledge of the domain, and experience with the limitations and demands placed on the systems they are developing. This knowledge enables them to better assess the quality characteristics of the complete product.

A practical example is that of a mechanical arm: the hydraulic valves and electric motors are all controlled by a micro-controller. The software for this micro-controller is often developed ad-hoc for each application, so no generalized test cases can be developed.

In the example here, we are developing tests for a module of the micro-controller software. The system under test (SUT) is a filter that ensures that a signal, e.g. the input for a motor from the user or other modules, does not damage other components. It does so by ensuring that the signal does not exceed a given upper limit and attenuating any sudden changes. This filter is a relatively simple, but typical component: it is common enough to be included in the basic function library that is provided with the development tool mentioned above.

To gain a better understanding of the situation, we conducted discussions with our industrial partner and attended a training session. Their approach is to provide domain specialists with training in the use of the software development tool, rather than trying to teach software engineers the domain concepts they would need. The software development tool they provide uses concepts that are already familiar to those working in the field, e.g. it expresses a component in terms of signals and operations on signals, rather than programming concepts. The trainees, in effect, create the types of models that they have been used to develop, and those models are used by the tool to generate code.

We have developed an Interactive Search Based Software Testing system, tailored to the specific need of the industrial partner and their development tool, and based on the type of software applications that they usually develop. The quality criteria we have used to evaluate this system and the resulting test cases are based on our discussions with our industrial partners, their current practices and their experiences. The ISBST system itself is a web-based add-on, separate from the tool offered by our industrial partners. This means that, while the current SUT and application are specific to the company, the ISBST system can be extended to address other software problems. This will allow any findings to be generalized to a wider set of problems and situations.
4.4 The ISBST System

The ISBST system, described in more detail in [60], consists of a search-based software testing system, where domain specialists can interact with and direct the search for test cases for a SUT. The actual SUT being tested is the source code generated from a visual model that the practitioner specifies with their existing IDE; the models are similar to function block diagrams used in the development of embedded control systems and software. The ISBST system then searches for test cases that violate desired requirements of the SUT (‘do not exceed this output signal limit’, and ‘avoid sharp discontinuities in the output signal’, respectively) while keeping test cases as short as possible.

The interaction allows domain specialists to use their experience to evaluate and rank the solutions developed by the ISBST system. To allow this interaction to take place in a meaningful way, the system uses representations that are relevant to the domain specialist, combined input and output signal graphs as well as tables of input and output pairs. By staying close to the way that the practitioners currently think about and specify test cases there is less of a gap in using the ISBST system. This type of domain specific representation, separating the domain specialist from the minutiae of the underlying ISBST system, e.g. they do not need to care about how to represent the test cases to be amenable to search etc.

This means that the specialists in question will continue to work with concepts and ideas familiar to them and do not need to develop software-specific skills to use the ISBST system to create test cases.
4.4 The ISBST System

The ISBST system (Fig. 4.1) consists of two nested cycles. The inner cycle contains the search based software testing system that forms the basis of our approach. The domain specialist guides the search indirectly, by selecting the quality criteria that they consider important at a given time and prioritizing them. The fitness function is then adapted to reflect those priorities. The outer cycle handles the interaction with the domain specialist, including quality criterion selection and prioritization, and the visualization of the candidates.

On a more technical level, the inner cycle is the back-end of the system and has been developed in Ruby. It uses the differential evolution algorithm found in the FeldtRuby library to evolve the candidate solutions, and then a variant of Bentley’s Sum of Weighted Global Ratios [50] method for evaluating them.

The outer cycle uses combination of html and javascript to display the candidate solutions to the domain specialist. The Data-Driven Documents library (D3) to provide a suitable graphical visualization for the candidate solutions.

As mentioned above, Takagi [10] identifies user fatigue as a main problem in using interactive evolutionary computation. He also proposes some methods of alleviating that problem.

In our system, the outer cycle handles any issues regarding the interaction with the domain specialist, including the prevention of user fatigue. To this end, some of the methods presented in [10] have been adopted. Only some of the large number of solutions being generated are displayed, with the selection being performed on the basis of the priorities the domain specialist has stated. Interaction events take place once every 500 optimization steps, to further alleviate the problem of fatigue.

4.4.1 Use of the ISBST System

The current ISBST prototype is used via a web interface. It allows the domain specialist to select from a list of objectives those that are relevant to them currently and to provide a prioritization by assigning the appropriate weight to each objective.

The weighting provided is used to dynamically develop the Intermediate Fitness Function (IFF) that will be used by the ISBST to create the first set of candidate solutions.

During the next interaction step, the domain specialist can evaluate the top ranked solutions from that set. An overview of the candidate solutions allows for an at-a-glance comparison (Fig 4.2), while individual views can provide additional information.
Practitioner-Oriented Visualization in an Interactive Search-Based Software Test Creation Tool

Figure 4.2: An overview of the best available candidate solutions during an interaction. The diagram shows information regarding the score each candidate solution obtained regarding the three objectives that were used for the evaluation: Length of the test case (Y axis), the degree to which each test case approaches or even exceeds a set upper limit is given by the Upper Limit (X axis) score, and the number of discontinuities that were discovered (color). This diagram provides an at-a-glance way of comparing candidates. More information on each candidate is also available in the form of an individual view.

It is also during the interaction step that the domain specialist to adjust their selection and prioritization of the objectives by re-weighting them. Once all these activities are complete, the process of dynamically computing the IFF, developing a new set of candidate solutions and presenting the top ranked ones for evaluation is repeated until one or more satisfactory candidate solutions have emerged.

4.5 Empirical Evaluation

The empirical evaluation of the ISBST prototype has two goals: the wider one of investigating the applicability of ISBST for companies in this industrial domain, and the narrower evaluation of the interaction mechanisms chosen for the prototype. These goals constitute an initial evaluation, part of the wider goal of determining what is the level of quality of the tests found using this method. An overview of the evaluation method can be seen in figure 4.3.

The first goal, the applicability of ISBST in this industrial context, was evaluated by the degree to which the prototype system evolved solutions according to the search objectives and weightings set by the domain specialist.
The second goal, evaluating and improving the interaction mechanism used by the ISBST prototype system. The mechanism is that of dynamically developing the Intermediate Fitness Function (IFF) based on the objective selection and re-weighting. To evaluate this mechanism, we measure the degree to which the diversity of the resulting test cases is affected by the choice of fitness function.

The empirical evaluation consisted of two stages. In the first stage, the choice of interaction mechanism was validated in a laboratory setting. The second stage consisted of evaluating the ISBST prototype with the company’s development and testing team. In practice, the two stages overlapped into a Continuous Development Stage (fig. 4.3), with new information being incorporated into the system.

The second stage consisted of workshops and interviews with the development and testing team. These workshops provided further validation for the interaction mechanism, as well as lessons regarding improvements that can be made to the system.

The workshops also resulted in a set of lessons learned, described in Table 4.2.
Practitioner-Oriented Visualization in an Interactive Search-Based Software Test Creation Tool

Table 4.1: Interaction strategies used to investigate the preservation of population diversity

<table>
<thead>
<tr>
<th>Scenario</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>Non-dynamic, multi-objective fitness function.</td>
</tr>
<tr>
<td>B</td>
<td>Focus in on one single objective that completely and consistently outweighs the others.</td>
</tr>
<tr>
<td>C</td>
<td>The objective under focus outweighs the others, but is changed at the last step.</td>
</tr>
<tr>
<td>D</td>
<td>A more balanced approach. One objective outweighs the others, but it is not the sole focus of attention.</td>
</tr>
</tbody>
</table>

Figure 4.4: Population diversity after a set of Interactive Objective Re-Weightings
4.6 Discussion

Table 4.2: Lessons learned as a result of the empirical evaluation

<table>
<thead>
<tr>
<th>Lesson</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td><strong>Understandability.</strong> Providing clear, understandable, and accurate information to the domain specialist is the key element in establishing a meaningful interaction between them and the ISBST system.</td>
</tr>
<tr>
<td>2</td>
<td><strong>Variation.</strong> A single developer may work on several systems in a short amount of time. In addition, there is a large amount of variation in terms of the types of systems developed within the same company.</td>
</tr>
<tr>
<td>3</td>
<td><strong>Individual Preferences.</strong> Even when faced with similar systems, different individual domain specialists may have different preferences regarding how the interaction should take place and what information should be made available.</td>
</tr>
<tr>
<td>4</td>
<td><strong>Dynamic Validation.</strong> Dynamically validating the system allowed us to identify potential problems in a timely manner and to develop a system that was relevant to our industrial partners and that could, in turn, benefit from their experiences.</td>
</tr>
</tbody>
</table>

Overall the results indicate that the mechanisms we have used to handle the interaction between the domain specialist and the ISBST prototype are useful and usable. The domain specialists responded well to the prototype and were extremely helpful both in evaluating it and in providing suggestions for improvements in subsequent versions.

On a more practical level, a number of the tests generated were successful in achieving the search objectives, i.e. identifying inputs that cause the system of exceed the maximum value set or cause the output signal to have discontinuities.

4.6 Discussion

For future work, one medium term goal is to tap into the knowledge and experience of domain specialists to guide the ISBST system towards interesting and meaningful test cases, thus improving quality without leading to prohibitive costs. Another is to apply this concept in other domains that can benefit from automated test case development, yet where domain knowledge is vital. Further
still, other phases of the software development process could also benefit from
the combination of human insight and automated computing power.

In the long term, we hope to enable human inspiration and experience, things that cannot be captured in an automated system, to guide rather than limit software development. Such human specific contributions would, therefore, actively improve the solutions rather than being abstracted away as inconvenient or unpredictable.

In addition to this vision, however, some threats to the validity of this study must also be discussed.

The ISBST prototype system was developed for a specific company and is therefore limited by using a single set of procedures and relying on one source of information and experience. That said, the company develops a wide range of embedded software for a generic type of controller. Future studies will investigate the issue of generalizability further, but, based on current results, we feel that this approach shows considerable promise.

4.7 Conclusions

This paper has presented an industrial application of search based algorithms. The Interactive Search Based Software Testing (ISBST) prototype system we have proposed and implemented uses the knowledge and experience of domain specialists to guide the search algorithms towards interesting solutions.

Initial results are promising, showing that the interaction between domain specialists and automated test case generation tools is a sound approach and can yield useful results. The same results also show the importance of the interaction mechanisms and of the information being provided to the domain specialists, in order to make their decisions.

On a higher level, the importance of dynamic validation, with company personnel and in an industrial context is apparent, as practical use of the prototype yielded more information that the preceding static validation efforts.
Chapter 5

An Initial Industrial Evaluation of Interactive Search-Based Testing for Embedded Software

Bogdan Marculescu, Robert Feldt, Richard Torkar, Simon Poulding

5.1 Introduction

Software, especially embedded software, is an essential part of a variety of complex systems that are used in many domains. The companies developing such systems focus their core competencies on domain specific knowledge and experience, rather than software engineering and software testing. As a result, they often lack the expertise to perform systematic software testing and quality assurance, focusing instead on testing the product as a whole. Since the quality of the developed products depends on a series of trade-offs, software quality assurance is often not a priority concern. Developing in-house software expertise
is prohibitively expensive and companies often prefer to focus their resources on improving domain specific competitive advantages. 1

It therefore becomes important to enable domain specialists to improve the quality of the software they develop, without shifting their focus away from their primary concerns. This could be achieved by developing a pre-packaged software testing toolkit that would offer the best practices in software development without the need to master the details behind the tool. This concept is sound, but developing such a package before the specifics of the application become known is a difficult task. Moreover, the functionality of the applications and the ways they are tested may change, or may differ between different testers and domain specialists involved, further emphasizing the importance of being able to use domain knowledge as an integral part of the testing process and have a flexible tool that can adapt to different scenarios and types of usage.

This paper proposes a system for testing embedded software by applying a technique that largely automates the generation of test data while still enabling domain specialists to contribute their knowledge and experience, thus allowing them to focus on domain-specific concerns. The automated technique applied uses a metaheuristic optimization algorithm to generate the test data and thus is a form of Search-Based Software Testing [8, 9]. The domain specialist interacts with the system to guide the optimization algorithm in the generation of test cases that are appropriate in a given context. This interaction is inspired by existing work in Interactive Evolutionary Computation [10, 12, 27, 28, 36], and is designed to make it easy for the domain specialist to make their contribution, while shielding them from the implementation details of the tool itself.

The contributions of this paper are as follows:

- A proposal that search-based software testing may be combined with user interaction with the objective of permitting test cases to be generated efficiently by users who are not necessarily testing experts.

- A description of how this proposal was implemented as an Interactive Search-Based Software Testing (ISBST) system during a case study in collaboration with an industrial partner.

- An industrial evaluation demonstrating that the ISBST system can be successfully be used by domain specialists to develop test cases without requiring extensive training in its use.

---

1 We will use the phrase ‘domain specialists’ to describe system engineers and other specialists who use, develop and test software, even though their focus is firmly on their particular domain.
• A laboratory experiment that validates the contribution of the underlying search-based test generation algorithm. This experiment compares the effect of the algorithm in the context of different interaction strategies that are based on data gathered during the industrial evaluation.

In Section 5.2, we consider existing approaches to interactive evolutionary search, and discuss how our approach differs from them. Section 5.3 is an overview of the industrial case study, and Section 5.4 describes the ISBST system developed during the study. Section 5.5 describes an evaluation of the system by users from our industrial partner. A laboratory experiment motivated by the results of the evaluation is described in Section 5.6. The results of the industrial evaluation and laboratory experiment are discussed in Section 5.7. Threats to validity are discussed in Section 5.8. Section 5.9 concludes the paper.

5.2 Related Work

Search-Based Software Engineering (SBSE) is a term coined by Harman and Jones in 2001 [7] to describe the application of metaheuristic optimization (or ‘search’) algorithms to software engineering problems, see e.g. [11, 12, 13]. The branch of SBSE concerned with testing problems is known as Search Based Software Testing (SBST) and has been applied to many types of testing problems [8, 9], from object-oriented containers [24] to dynamic programming languages [25].

The premise of SBSE is that for many software engineering problems it is difficult to derive a solution directly, but it is often easy to check whether a given ‘candidate’ solution solves the problem. In the context of SBST, the problem is typically to derive test data that satisfies a specific testing objective: while it may be difficult to derive a suitable test case, if we are given a candidate test case it is usually straightforward to check whether it meets the testing objective. If a fitness function can be defined that measures the extent to which the candidate solution solves the problem, then it is possible to use this fitness function to guide a metaheuristic optimization algorithm towards solutions that solve the problem. Even though the optimization algorithm may need to construct and evaluate a large number of candidate solutions to find one that solves the problem, this approach is often less costly than solving the same engineering problem manually. Many metaheuristic optimization algorithms operate on a population, i.e. a set of individual candidate solutions, and such an algorithm is used in the ISBST system described in this paper. For this reason, we will use the terms ‘candidate solution’, ‘candidate’, and ‘individual’ interchangeably to refer to a potential solution developed by a search-based system.
There have been comparatively few studies considering interactive SBSE or SBST. Feldt [27] described an interactive development environment where tests are created as the engineer writes the program code or refines the specification. The system used the interactions of the engineer to help guide the search but the effect on the fitness function was indirect. Other work by Feldt [12], and by Parmee et al. [59], considered the use of interactive search to explore engineering designs and better understand design constraints but did not focus directly on software testing.

Nevertheless, the notion of interactive involvement in a search process is well-established. Takagi describes Interactive Evolutionary Computation (IEC) as “an EC that optimizes systems based on subjective human evaluation” [10]. This approach uses the human as a replacement fitness function in situations where the optimization goal is dependent on “human preference, intuition, emotion and psychological aspects” [10]; this includes applications such as arts and animation, computer generated graphics and image processing.

Takagi [10] also identifies three main approaches to human interaction with an Evolutionary Computation (EC) system. First, the human can act as a regular fitness function: the human is presented a set of candidates and must assign a fitness score to each of them. This means that each candidate must be analyzed and evaluated.

The second approach is to present the human with the candidates to be evaluated; the human then chooses those that are remarkable, either selecting the ‘good’ candidates for promotion to the next generation or selecting the ‘bad’ ones for exclusion. Only a subset of candidates need to be marked, ranked or graded, in this approach. This helps guide the search by ensuring that desired characteristics are always represented in the population and have a higher chance of propagating to the next generation. In effect, the user guides the search by selecting those candidates deemed to be the “best current representation of the goal” [37].

The third approach identified is that of Visualized EC, where the human selects a solution based on the fitness values for several objectives, rather than analyzing the individual candidates themselves. The approach is described in more detail in [38]. One such example is presented by Bavota et al. [39], where a candidate solution is a proposed distribution of software components into clusters. Rather than evaluating each candidate itself, the user is required to decide if two components belong in the same cluster or not.

One additional concept related to IEC is that of hyper-interactivity, defined as a “form of IEC in which a human user actively chooses when and how to apply each of the available evolutionary operators, playing the central role in the
control flow of evolutionary search processes” [36]. Here, the human acts as a direct guiding hand into each candidate’s development rather than as a substitute fitness function, evaluating candidates after they have been generated.

All these approaches require that the human interact with the system at least once each generation. The first two approaches imply that the human should assess and evaluate each candidate before assigning fitness scores or making their selections. These candidate individuals may be quite complex constructs, leading to difficulties in making consistent, impartial and accurate evaluations.

By having the human analyze each candidate, to a greater or lesser degree, the number of candidates that can be processed is reduced. The problem of decision fatigue has already been identified and efforts have been made to address it or, at least minimize its effects [61].

One proposed way to address the problem of human fatigue, especially decision fatigue, is that of finding a way of measuring fitness that does not require the user to interact with the system so frequently. For example, Tonella et al. describe a system that requires user input only when the existing fitness function prioritization results in a tie [64], thereby decreasing the demands on the human user.

The Visualized EC approach (described above) requires only that a human evaluate the fitness scores that each candidate has already received, rather than the candidate solution itself. This, however, still means that each candidate solution must be considered and compared to the others. Hyper-interactivity means that the human user should be extremely involved not just with evaluating each candidate, but with developing candidates as well, applying the evolutionary operators.

An alternative approach, that is quite different from those described so far, is that of developing surrogate fitness functions aimed at standing in for human behavior. Chou et al. [65], and Sun et al. [66] go into greater detail on how this would be achieved. While this approach does resemble our purpose, the key difference is that the quality criteria we are investigating can suffer dramatic changes as new information becomes available. In our context, the human subjective evaluation is a way for domain specialists to contribute their knowledge and experience, while still keeping the focus firmly on objective quality criteria rather than aesthetic ones.

Closer to our context is the work of Liapis et al. [67], where a user selects their preferred solution and the system re-weights the quality objectives until the user-preferred solution has the highest fitness. In contrast, our goal is to actively work to ensure that the fitness function is a close match to the domain specialist’s current understanding of the priorities and relative importance of
the quality objectives. Thus, even if the candidates currently displayed do not exhibit the qualities the domain specialist requires, they can still guide the search according to their estimation of what quality criteria the system should fulfill.

Avigad and Moshaiov [68] combine computable performance with decision maker preference to improve the process of selecting a conceptual design. Their work, however, focuses on a problem where performance can be objectively computed, and the criteria for doing so are already in place. The decision maker’s influence is, ideally, to choose between designs comparable in quality. Deb et al. [69], similarly use user preference as an input, in a “progressively interactive” manner, on a set of non-dominated points. Their paper implies that there exists a set of objective measures to determine dominance and a set of criteria by which dominance is judged. By contrast, our work seeks to explore the space of test cases and find “interesting”, i.e. not previously known, test cases and behaviors. An objective performance measure is hard to define in this context and the criteria for what constitutes such an interesting solution are fluid.

Simons and Parmee [70], define elegance as a key factor in software design. They define a set of quantitative measures for elegance, and their fitness evaluation takes one of those measures, randomly, and shows the user the most elegant solution according to that measure. These measures are quite specific to the domain and the authors’ definition of elegance. In our context, the domain specialists’ understanding of what constitutes a good solution may vary with time, or might not be so quantifiable to begin with.

In a previous paper [60], we proposed a system that combines several of these concepts, e.g. interacting with the system once in a number of generations, rather than each generation [61]; and adds that of interacting with an ISBST system by means of allowing the human to modify the fitness function dynamically during the search process as their understanding changes or becomes more refined, or as new information becomes available. In this paper we seek to expand on that work and evaluate, in an industrial setting, both the utility of the ISBST approach and how users interact with the search process.

5.3 Case Study Overview

The implementation of ISBST presented and evaluated in this paper is a result of a case study undertaken in collaboration with an industrial partner. This section
5.3 Case Study Overview

describes the context in which our partner operates, provides an overview of the study design, and identifies two key research questions.

5.3.1 Industrial Context

Our industrial partner (who we are unable to name for reasons of confidentiality) develops off-highway vehicles and components: Products that involve embedded software, but where software is not the main consideration. As a result, knowledge of the domain and domain specific trade-offs are critical to quality. This requires a domain specialist to assess quality characteristics that the complete product must have, rather than focusing on the software components alone.

To enable their customers to develop their own products and to adapt components to their own particular application, our partner also provides a graphical programming environment, which we will refer to as ‘DomainDevEnvironment’, that uses a drag-and-drop interface to create control systems. This environment allows their clients’ engineers to use concepts they are familiar with to create software for the partner’s micro-controllers. The code that will be deployed to those controllers is automatically generated from the graphical designs that the engineers produce.

To better illustrate the context, consider the examples of a control mechanism for an electric motor powering a mechanical arm. Due to limitations of the motor itself and the potential for damage in what the mechanical arm is handling, a SoftRamp component is necessary to ensure that sharp increases or decreases in the input signal do not damage the motor.

The users of ISBST system are therefore domain specialists who use the DomainDevEnvironment toolkit and may come from different, albeit related, backgrounds and are developing a wide variety of different products.

5.3.2 Research Questions

This paper focuses on two research questions that may be addressed by the case study:

1. What is the domain specialists’ evaluation of the ISBST system in terms of usefulness and usability?

2. How effective is the interaction between the domain specialists and the ISBST system?
The first question is a qualitative evaluation of the ISBST system, from the perspective of its intended users.

To answer the second question, we first determine if the necessary information flows appropriately between the system and the domain specialist. The interaction is deemed effective if the domain specialist has the information needed to make an informed decision, and subsequently provides enough information back to the system to guide the search further.

5.3.3 Study Design

The case study has been conducted over a period of two years, and covers both the development and evaluation the ISBST system in collaboration with our industrial partner. This long-term involvement with an industrial partner is critical at an early stage in the development of interactive search-based systems; our overall approach is an example of design research [71]. The study design is summarized in Figure 5.1.

Initial information was obtained from workshops with the company and by attending training sessions. Subsequent development of the ISBST system was guided and validated by frequent discussions, workshops and participation in company training sessions. In particular, context-specific quality objectives for test cases in this implementation are derived from discussions with our industrial partner and their clients, frequently updated according to their perceptions, and based on their experiences. Examples of these quality objectives are identified in description of the ISBST in this paper, but the general ISBST approach is not limited to these specific objectives. The ISBST system developed by this collaboration is described in Section 5.4.
At the end of development, the ISBST system was validated on-site, with domain specialists from the company, testing production code under realistic conditions. This evaluation is described in Section 5.5.

During the evaluation we observed different “interaction strategies”, i.e. patterns in the interaction between the system and the domain specialists. These strategies were subsequently used to conduct a laboratory experiment to determine how the interaction strategy affects the results of the underlying search-based system. This experiment is described in Section 5.6.

The results of the industrial evaluation and laboratory experiment will be used to answer the two research questions identified above.

5.4 The ISBST System

This section describes the ISBST system which was developed in collaboration with our industrial partner.

While the general approach of ISBST has broad applicability, this description also covers the implementation-specific details. In particular, the exemplar system under test (SUT) is SoftRamp, a common component in the DomainDevEnvironment toolkit used by our partner and whose functionality was described above; and the set of quality objectives is specific to the type of software that is developed using the toolkit.

5.4.1 Overview

The ISBST system is designed to make it easy for the domain specialist to interact and contribute their knowledge and experience, while at the same time shielding them from the minutiae of the underlying search-based system.

To achieve this goal, the system can be imagined as two nested cycles, as seen in Figure 5.2. The inner cycle uses a search-based algorithm to create a population of candidate solutions. The outer cycle is concerned with interacting with the domain specialist and converting their feedback into the appropriate fitness function that guides the search algorithm of the inner cycle.

The system maintains the overall structure presented in our previous work [60], while making allowances for the increasing complexity inevitable in practical implementation.
Figure 5.2: Overview of the ISBST System
5.4 The ISBST System

5.4.2 The Inner Cycle

The inner cycle contains all the technical scaffolding needed to develop candidate solutions and to interface with the SUT in order to assess quality objectives.

Encoding a Solution. A candidate solution in our context is a test case consisting of a set of inputs, generated by the search-based algorithm.

For the SoftRamp component, the inputs are divided into setup values and input signal. The setup values are a set of five integers that determine the setup of the SoftRamp. Some of these values are subject to special rules, e.g. two values, $sft_{strt}$ and $sft_{end}$ are percentages and $sft_{strt} + sft_{end} \leq 100$. These values are randomly generated from the acceptable values. The input signal can consist of any datatype supported by the DomainDevEnvironment. The SUT currently used accepts an input signal with a fixed length of 15 values, each a 16-bit integer.

So, a candidate solution is a vector of 20 real numbers, the first 5 being the setup values for the SUT, and the remaining 15 being the input signal for the SUT. An output signal is generated for each candidate solution by running the SUT with the 5 setup values and collecting the output signal that matches the input.

Quality Objectives. During the study, we identified a number of “quality objectives”, i.e. characteristics of a test case that would indicate a problem in the SUT and, thus, make a good test case. These quality objectives, described in more detail in Table 5.1, are calculated from the outputs of the SUT when run with the candidate test case.

Intermediate Fitness Function. The mechanism we chose to allow the domain specialists to guide the search is to encode their feedback as to the relative importance of the quality objectives in the Intermediate Fitness Function (IFF). This function is calculated from of a set of quality objective scores for a candidate and a set of weights for those objectives, and is a variant of Bentley’s Sum of Weighted Global Ratios [50]. This approach normalizes all the values in a generation to an interval between the largest and the smallest values observed for a given objective, both in the current and previous generations. Each solution is assessed and receives a score for each of the quality objectives. The weights are then used to combine the scores into a single fitness value for each candidate.

\[
IFF(j) = \sum_{i=1}^{n_{	ext{Objectives}}} \text{Weight}_i \times \text{Value}_{i,j}
\]  

(5.1)
<table>
<thead>
<tr>
<th>No.</th>
<th>Objective</th>
<th>Description</th>
<th>Aim</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Large Signal at Least Once</td>
<td>Rewards higher maxima values in the output</td>
<td>Maximize</td>
</tr>
<tr>
<td>2</td>
<td>Small Signal at Least Once</td>
<td>Rewards lower minima in the output</td>
<td>Minimize</td>
</tr>
<tr>
<td>3</td>
<td>Mean Output</td>
<td>Rewards higher mean values in the output</td>
<td>Maximize</td>
</tr>
<tr>
<td>4</td>
<td>Above Important Limit at Least Once</td>
<td>Rewards test cases with output values that exceed by the most a prescribed maximum value. Only the largest difference is considered.</td>
<td>Maximize</td>
</tr>
<tr>
<td>5</td>
<td>Below Important Limit at Least Once</td>
<td>Rewards test cases with output values that fall farthest below a prescribed minimum value. Only the largest difference is considered.</td>
<td>Maximize</td>
</tr>
<tr>
<td>6</td>
<td>Above Important Limit Overall</td>
<td>Rewards test cases with mean output values that exceed by the most a prescribed maximum value.</td>
<td>Maximize</td>
</tr>
<tr>
<td>7</td>
<td>Below Important Limit Overall</td>
<td>Rewards test cases with mean output values that fall farthest below a prescribed minimum value.</td>
<td>Maximize</td>
</tr>
<tr>
<td>8</td>
<td>One Large Increase</td>
<td>Large variations in the output signal may damaging components, or be indicative of internal faults in the module. Rewards the maximum value of the first order derivative.</td>
<td>Maximize</td>
</tr>
<tr>
<td>9</td>
<td>One Large Decrease</td>
<td>Large variations in the output signal may damaging components, or be indicative of internal faults in the module. Rewards the minimum value of the first order derivative.</td>
<td>Minimize</td>
</tr>
<tr>
<td>10</td>
<td>Swings Through Zero</td>
<td>Oscillations in the output may be damaging to the other components or indicative of internal faults. Rewards the highest number of times the output signal crossed the 0 value.</td>
<td>Maximize</td>
</tr>
<tr>
<td>11</td>
<td>Diversity</td>
<td>This is an overall measurement of how different a test case is from the population in the previous interaction event. This allows the domain specialist to widen the search space being explored or converge towards a solution.</td>
<td>Maximize</td>
</tr>
</tbody>
</table>

Table 5.1: An overview of the current Quality Objectives

where $IFF(j)$ is the fitness value of candidate $j$, $Weight_i$ is the current weight of the objective $i$, and $Value_{i,j}$ is the fitness value of candidate $j$ measured by objective $i$. The value of $IFF(j)$ is the sum of the weighted fitness values for all $nObjective$ objectives. An objective $k$ can be deselected from the computation by having $Weight_k = 0$.

The weights are received from the Outer Cycle. The weights change according to the feedback received from the domain specialist, and the IFF is recomputed to reflect those changes. More detail on the quality objectives and how they relate to user interaction can be found in Subsection 5.4.3 below.

**Search Algorithm.** The search-based algorithm chosen for this implementation was Differential Evolution (DE) [49].
5.4 The ISBST System

Differential Evolution is a parallel direct search method. Each potential solution is a vector of real numbers. The initial population is chosen randomly from a uniform distribution, and covers the entire parameter space. New parameter vectors are added by mutation: adding the weighted difference between two population vectors to a third vector. For each target vector $x_{i,G}$, where $i = 1, 2, ..., NP$ a mutant vector is generated as follows:

$$v_{i,G+1} = x_{r_1,G} + F \times (x_{r_2,G} - x_{r_3,G})$$ (5.2)

where $r_1, r_2, r_3 \in 1, 2, ..., NP$, are integer, and mutually different and different from the running index $i$. $F$ is a real and constant factor $\in (0, 2]$ which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$.

The result $v_{i,G+1}$ is then subjected to crossover, by mixing its parameters with those of another predetermined vector, and the outcome of this operation is called trial vector. If the trial vector is an improvement over the target vector, it replaces it in the following generation [49].

The crossover rate we used is $cr = 0.5$, the scale factor is $F = 0.7$, and the population size is $population = 100$. The mutation strategy is that proposed by Storn and Price [49]: DE/rand/1/bin. The strategy uses a differential evolution algorithm (DE); the vector to be mutated is randomly chosen (rand); one difference vector is used (1); the crossover scheme is binomial (bin).

5.4.3 The Outer Cycle

The Outer Cycle is the component responsible for handling the interaction with the domain specialist.

**The Interaction with the Domain Specialist.** After a number of optimization steps, $steps = 300$, the Inner Cycle stops the search and triggers an “Interaction Event”. This event consists of: a) displaying the current and previous generations to the domain specialist; b) displaying additional details on demand; and c) allowing the specialist to guide how the search is to be continued.

The current and previous generations are displayed as graphs, with the graph axes showing the fitness values obtained by each candidate solution with respect to selected quality objectives, as can be seen in Figure 5.3. To visualize more than a pair of quality objectives, a domain specialist can choose to view a matrix of plots, where each graph in the matrix shows the fitness values of a pair of the selected quality objectives, e.g. Figure 5.4.
Figure 5.3: Single Scatterplot example. The current generation is shown in light blue; the previous generation is in orange. (Screenshot from the ISBST tool.)

A user can select one or more of the candidates being displayed and see a more detailed view. This view includes, in addition to the fitness values with respect to each objective, the input and output values, see Figure 5.5.

To guide the search, the domain specialist decides the relative importance of the quality objectives using the Guidance Panel in Figure 5.6. The relative importance is expressed in terms of weights, all of which have values between $weight_{min} = 0.0$ and $weight_{max} = 1.0$, and are modified in increments of $weight_{\delta} = 0.1$. The values do not have to add up to any fixed value. These weights are then used to compute the IFF, as described in Subsection 5.4.2 above.

At any point, the user is able to export test cases that they feel are particularly interesting for use and investigation outside of the ISBST system.

### 5.4.4 System Implementation

The outer cycle uses a combination of CoffeeScript$^2$, a variation of JavaScript aimed at simplifying development, and the Data-Driven Document (D3) library$^3$

---

$^2$http://coffeescript.org  
$^3$http://d3js.org
Figure 5.4: Scatterplot Matrix example. The current generation is shown in light blue; the previous generation is in orange. (Screenshot from the ISBST tool.)
Figure 5.5: A detailed view of one of the test case candidates. (Screenshot from the ISBST tool.)

Figure 5.6: Search Guidance panel. (Screenshot from the ISBST tool.)
to display candidates and obtain feedback from the domain specialist. D3 is a
JavaScript library for the manipulation and display of data in a browser. It
allows visualizations to be dynamically created, without acting to change the
data itself. Since the outer cycle is concerned with displaying the candidates,
the combination of Coffescript and the D3 library is a good fit for the purpose.

The inner cycle and the SUT interface are developed in Ruby. This was
chosen for the relative ease with which it interacts with other applications, e.g.
SUTs, interfaces to simulators or interfaces to hardware test benches, enabling
the ISBST to be more extensible overall.

Communication between the inner and outer cycles is achieved by packaging
candidate objects into a JavaScript Object Notation (JSON) file\(^4\) and made
available through the Sinatra\(^5\) framework. Search-related meta-information,
e.g. fitness values for each objective, is included in the candidate object, and
therefore available in the outer cycle. This enables the system to more easily
adapt to changes in display requirements.

5.5 Industrial Evaluation

This section describes an on-site evaluation that was conducted with our partner
company’s engineers.

5.5.1 Methodology

The objective of the evaluation was to observe how the engineers interact with
the ISBST system and to determine if the system can be successfully used to
develop relevant test cases.

The evaluation took place over the course of a single day at our industrial
partner’s site in Sweden and involved five of the company’s engineers. The
participants had diverse backgrounds, education and previous experience, as
can be seen in Table 5.2. All the participants were developing software using
the DomainDevEnvironment or were working on developing the tool itself. As a
result, they can all be considered to be domain specialists, due to their current
work with domain specific tools, as well as their previous experience in domain
specific activities.

The evaluation began with a brief presentation to clarify the purpose, and
how the case study relates to the company and its activities.

\(^4\)http://www.json.org
\(^5\)http://www.sinatrarb.com
The subsequent phases of the evaluation were as follows:

1. Collecting demographic information (5 minutes). An initial set of five questions was used to determine the subject’s previous experience, both in their position and with the SoftRamp SUT, as well as education background. For those that took part in previous validation efforts, impressions on those were also collected, as they could affect the current evaluation.

2. Understanding SBST and the ISBST prototype (10 minutes). This section contained a brief demo of the ISBST system, its operation, answers to any questions the subjects had, and clarifications to any of the practical issues regarding the use of the ISBST system.

3. Current testing procedures (10 minutes). In this phase the candidate discussed current test case development strategies and establish a baseline for the type, number and quality of test cases that the developer could create manually (i.e. without the assistance of the ISBST system).

4. Practical evaluation (15 minutes). During this phase, the subject used the ISBST system to develop test cases for the SoftRamp component. The participant had the freedom to choose the ultimate goal of the activity, so that the comparison to their regular testing procedure would have a common basis. No further answers nor clarifications were available during this phase. The interaction between the subject and the search-based system was logged, including the weights assigned to each objective for each interaction step. Further notes on the preference for certain strategies or objectives as well as any other information deemed interesting was also recorded by the researcher.

5. Debriefing and final questions (5 minutes). This phase consisted of a brief interview to collect feedback regarding the system, impressions regarding the interaction and the resulting test cases, and gave the subject the opportunity to provide any additional comments or suggestions.

The same procedure was followed for each of the participants in the experiment. Any departure from this procedure was noted. If the departure from the procedure was severe, then the data provided by the participant in question was eliminated from the analysis. This situation occurred in only one instance: participant 3 was interrupted during the evaluation, so the data regarding that participant was not taken into account when performing the analysis.
Table 5.2: Demographic data for the participants to the industrial evaluation. The DomainDevEnvironment Team develops the environment itself, while the software teams uses it to develop software for company applications.

### 5.5.2 Results

This evaluation had two main goals: a) to evaluate if the information flow between the ISBST system and the domain specialist is sufficient for the latter to make informed choices and successfully guide the system; and b) to evaluate if the ISBST system can develop candidate solutions of comparable quality to hand-crafted solutions.

The interface received a largely positive evaluation, with respect to the first goal mentioned above. Though some issues were identified with respect to the clarity of the presentation of available information, the information itself was deemed useful and interesting. The information flow was clear and useful: that the participants were able to guide the search.

Figure 5.7 shows an overview of the answers to the final questions of the evaluation, after the candidates had had a chance to try the ISBST system out and use it to develop test cases.

First of all, all the participants were able to use the ISBST system without incident or additional support and develop test cases for the given SUT. The interviews revealed that 3 of the 4 participants would add some of the tests they developed using the ISBST tool to the regular test suite. All participants agreed that, even if the candidate test cases they had developed were not as good as the hand-crafted ones, they did seem to investigate different and important areas of the search space. Participant 1 stated that one of the test cases he had found, including large oscillations in the input signal, was “something a human tester may not think to check for”. The interviews also revealed that
differences of opinion are not uncommon, even between developers working on similar projects, in a similar context, and within the same company.

Overall, the ISBST system yielded good results: after only a short amount of training in its use, and within only 15 minutes effective working time, engineers were able to understand the test cases they were shown, reason about them, and successfully guide the search towards better candidates.

### 5.5.3 Interaction Strategies

One of the remarkable results of the evaluation was the variety of strategies the domain specialists used to interact with the system. This section will describe some of the different approaches and their implications.

The strategies discussed here emerged from three major information sources. First, the initial and final interviews contained questions regarding each participant’s approach to testing a given module. Second, the participants’ interaction with the system was closely observed, and the observations enriched the initial understanding of the participants’ approach to testing the SUT. Third, logs of the interaction events were collected, to provide a quantitative view of the participant’s approach to using the ISBST system. All the participants were informed about the duration of the evaluation phase, and it is reasonable to
assume that their interaction strategy was aimed at making the best possible use of the available time.

The first noticeable strategy consisted of isolating a subset of related quality objectives, based on experience. After the first interaction event, the participant then tried to focus on the candidates that had performed best overall and worst overall and subject them to closer analysis. During the discussion, the participant stated that their current approach to testing was to investigate extreme values. Their experience was that most faults emerged in those areas. A limited selection of test cases with intermediate values were checked, as a sanity check measure. In terms of interaction, this strategy resulted in a limited number of interaction events and small variations in the weights assigned to the various selected quality objectives. This is due to the participant focusing their attention and time on the quality objectives, trying to identify the objective combinations that resulted in greatest change in the test cases.

A second strategy was to focus on candidates that differed greatly in terms of one quality objective, but very little in terms of the others. Pairs of such candidates were identified and selected, and the detailed views of each candidate were used for conducting the comparison. The interaction was focused on analyzing the graphs of each pair of candidates and attempting to ascertain the reason behind the differences that were identified.

The third strategy involved more interactions with the system and looking at several generations of candidates. The weighting of quality objectives changed a lot during interaction events, with certain objectives being dropped altogether. The participant focused on the outlier candidates and looked for candidate solutions that exhibited unexpected characteristics. The use of the system was to investigate groups of inputs that were not normally included in manual test cases and check if any outliers exhibited unwanted behavior. The interactions between this participant and the systems were focused on the overview of the entire population, identifying outliers and looking at the differences between the current generation and the previous generation. This strategy was the one most in line with our expectations of how the system would be used.

In addition to the information provided by the different strategies, the participants’ approaches to interaction highlighted the variable nature of personal experience and expectations. It is important for a system that relies so heavily on interaction and the experience of its users to allow for variation in terms of how those users interact with it. This evaluation reinforces the importance of having a robust system that can interact with the domain specialists on their terms and according to their strategies. It also shows that the ISBST system has this flexibility and robustness.
5.6 Laboratory Experiment

This section describes a subsequent laboratory experiment that expands on the results of the industrial evaluation. It is motivated by a concern as to whether the search-based algorithm applied in the inner loop is effective for all the different interaction strategies observed during the evaluation.

5.6.1 Experimental Setup

First, four interaction strategies were defined based on the data obtained from the industrial evaluation. The most complex interaction observed in the industrial evaluation was chosen for the experiment. This complex interaction is the Realistic strategy, with only minor modifications to allow for a better comparison with other strategies. Several simplified versions of the Realistic strategy were developed to represent the range of different behaviors observed in the evaluation. These simplified strategies can be seen in more detail in Table 5.3. The Realistic strategy is the more complex and closer to actual domain specialist behavior, while the Null strategy mimics the behavior of a Search-Based Software Testing system with no interaction.

The Realistic strategy focused on optimizing a subset of 5 of the quality objectives: 1, 2, 8, 9, and 10 in Table 5.1. The selection of the objectives was based on the interaction logs, with the chosen objectives exhibiting both synergy and contradictions. To enable a fair comparison, all the simplified versions use the same subset of objectives.

Second, these strategies were compared by running the ISBST system with each strategy and measuring the overall fitness value of the population.

To ensure a fair comparison, the same “effort” was applied for each strategy and this was measured in terms of fitness evaluations made by the search algorithm, as recommended by Črepinšek et al [72]. All strategies are of the same length: 11 interaction steps, amounting to 3,300 fitness evaluations. Moreover, where the strategies call for changing the fitness function by re-weighting, the overall sum of the weights is the same, to prevent one of the strategies from obtaining an unfair advantage.

The ISBST system uses an initial random population and applies a default weighting in the Intermediate Fitness Function to create the population that a user sees at the first interaction event. The comparison is therefore made between the initial populations, and those at the end of 11 interaction steps.

The search algorithm is stochastic, and so, to minimize the effect of this randomness on the evaluation, each strategy is run a total of 30 times. To
### 5.6 Laboratory Experiment

#### Table 5.3: An overview of the Interaction Strategies used for the experiment

<table>
<thead>
<tr>
<th>No.</th>
<th>Strategy</th>
<th>Description</th>
<th>Comments</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Null Strategy</td>
<td>All the objectives have the same weight: 0.5.</td>
<td>It is the default strategy of the system and the control group of the experiment.</td>
</tr>
<tr>
<td>2</td>
<td>Clear Fitness Function</td>
<td>The 5 selected objectives have weight 1, the rest 0.1.</td>
<td>It simulates a case where the domain specialist knows from the first step what type of fitness function they are looking for. While not realistic, it is a plausible case and a useful comparison.</td>
</tr>
<tr>
<td>3</td>
<td>Slowly Finding a Fitness Function</td>
<td>The weights of the objectives change, but the change is gradual. All the 5 selected objectives start at weight 0.1 and slowly increase in priority until reaching 1.</td>
<td>This simulates a case where the domain specialist starts from a different weighting, and slowly adjusts it until finding the goal. Though not completely realistic, such behavior has been observed during the industrial evaluation.</td>
</tr>
<tr>
<td>4</td>
<td>Realistic</td>
<td>Each of the 5 selected objectives is the top priority during two interaction steps, the overall sums of their weights are equal, and no two objectives are top priority simultaneously.</td>
<td>This captures the most realistic interaction, while preserving equal overall weights and allowing a comparison with the other strategies.</td>
</tr>
</tbody>
</table>

facilitate this repetition, the strategies were be applied automatically by a script taking the place of a human in the Outer Loop.

#### 5.6.2 Results

A first analysis compared the initial population to the final one, using each of the strategies. Statistical significance between the initial and final populations was assessed using a paired Wilcoxon Signed-Rank Test over the sample of 30 results for each combination of objective and strategy. For the Realistic strategy, for objectives 1 through 5, the p-values were $p < 1e-06$. The comparison of the initial to the final populations yields similar values for all the strategies.

There was a significant difference between all the objective fitness values for all the strategies. This clearly shows that the search-based algorithm powering the inner loop clearly affects the outcome, regardless of the interaction strategy taken.

The box plots on the left side of Figure 5.8 compare the change in each of the five quality objectives when the Realistic strategy is applied. The figure shows a clear distinction between the initial and the final populations, thus indicating that the underlying search-based algorithm is performing as we assumed it
(a) Comparison of all the quality objectives for the Realistic Strategy.

(b) Comparison of quality objective o3 for all the Interaction Strategies. The objective o3 was randomly picked for display purposes, but all quality objectives show the same effect.

Figure 5.8: A detailed view of some of the experimental results.
Table 5.4: A comparison of the weights assigned to each objective at each step, for strategies S3 and S4. Note that the overall sum of the weights is equal, to ensure that all objectives a fair evaluation.

would. The goal for objectives 1, 3 and 5 was to maximize the fitness values, while the values objectives 2 and 4 were to be minimized, and the results in Figure 5.8 reflect these goals.

An interesting comparison is that shown in the right side of Figure 5.8, showing the change in the value of one quality objective (o3) across all of the interaction strategies. The displayed objective was chosen randomly, but all show the same behavior across the different strategies: The Null strategy performs by far worst of the group, while the other three strategies are much closer in terms of fitness values.

The comparison between the three remaining strategies, i.e. Clear, Slow, and Realistic, is quite straightforward. The Clear strategy shows a fitness function that never changes, yielding the best results, but assuming precise knowledge. The Realistic strategy shows slightly worse results, but starts from the most realistic assumption: that a domain specialist cannot accurately predict the desired fitness function from the start of the test.

In addition, a paired Wilcoxon Signed-Rank Test was conducted between the final population of the Realistic strategy and the Null strategy. The p-values for this test, for objectives 1 through 5 respectively, were $p < .001$, with the exception of the last objective, where $p = 0.6324$. The comparison is not significant with respect to the last objective, the number of times the output signal crosses zero. Due to the limitations of the current implementation, the values obtained for this objective varied between a small set of values. For all
the other objectives, an interactive strategy, even a sub-optimal one, clearly outperformed the non-interactive Null strategy.

Our assumption was that the Realistic strategy would create a more diverse set of solutions, but one that also had a significantly lower overall fitness value. The results of this experiment seem to indicate that the it performs better than expected. They also seem to show that the choice of interaction strategy is not critical to obtaining good results. Using a Clear or Slow strategy would be preferable, but if the available information does not allow such a choice, the Realistic strategy will provide good results nevertheless.

5.7 Discussion

Any discussion of the results should start with the statement that this was a case study on an ISBST system developed for a particular context and evaluated within one particular company. The case study consisted of an industrial evaluation and a laboratory experiment set up on the basis of the information obtained from the evaluation. As a result, it can only be claimed that these findings apply in this context, and conclusions more general than that cannot be drawn at this stage.

That said, we believe that the results indicate that ISBST is a potentially viable tool for use in industry. Domain specialists at our industrial partner were able, after a brief tutorial session and with limited time, to create interesting test cases. In some instances, the created test cases were comparable in quality to manually crafted test cases, and all the participants agreed that the tests were useful in terms of investigating behaviors that are not currently covered by tests, or provided insights into the workings of the tested (SoftRamp) module.

The current study has helped show some of the complexities of applying the ISBST system in an industrial context. Domain specialists have varied backgrounds and approach their duties in various ways, while search objectives may vary from one context to another. Nevertheless, search-based techniques can be applied in an industrial context and have, so far, yielded interesting results.

Next, we will discuss answers to the research questions we posed in Section 5.3.2.

1. What is the domain specialists’ evaluation of the ISBST system in terms of usefulness and usability?
The domain specialists that participated in our evaluation found the system usable and were able to develop test cases for the SUT. While improvements can still be made, all the participants were able to interact with the system and evaluate the results without incident.

The usefulness of the resulting test cases, and of the ISBST system itself, centered around its ability to develop test cases that were different from hand-crafted ones, in a short amount of time and with little additional training.

From this we can conclude that the ISBST system is usable, even with limited specific training. The results of the case study also indicate that the ISBST system is useful, as a complement to existing techniques. It provides a way of exploring, at a relatively low cost, areas of the search space that a human user would not otherwise investigate, but that may cause failures of the SUT.

2. **How effective is the interaction between the domain specialists and the ISBST system?**

The evaluation indicates that the interaction, as it is currently implemented, is sufficiently effective to allow the domain specialist to understand the test case candidates that the ISBST system generates, visualize in more detail those they find interesting, and make informed decisions that successfully guide the search for the next generation of candidates. All the participants in our case study were able to interact with the system, using diverse strategies, and achieve the goal of generating test cases for the SUT.

The laboratory experiment showed that the Null strategy, that mimics the behavior of a system with a fixed and a priori defined fitness function, performs consistently worse than all the other strategies (see Figure 5.8). This seems to indicate that the guidance obtained by interaction is an improvement over unguided search.

Overall, the evaluation of the ISBST system can be considered a success. The evaluation confirmed our assumption on the importance of domain expertise and validated our ISBST system.
5.8 Validity Threats

This section will discuss some of the validity threats identified with respect to this study. We will discuss these threats based on their root cause:

**Threats relating to sample size**  The case study is based around an industrial evaluation that included four engineers from our industrial partner. This is a small sample and this creates a number of threats regarding the degree to which the sample is representative, problems relating to random variances in the sample, and well as the generalizability of the study.

For the purposes of our study, we considered the population to be the total number of domain specialists employed with our industrial partner. The overall number of engineers that fit this requirement, together with the need for both domain expertise and experience with developing embedded software is extremely limited: i.e. less than 20 worldwide. As a result, our final sample of four engineers, while extremely limited, is representative of the population.

The sample size could be expanded, but only by including domain specialists from other companies. These additional participants would be developers of embedded software, but their domain expertise would differ enough from the original sample as to create additional threats to the validity of the study.

Overall, we claim that our conclusions are valid to the context of our industrial partner, though this study make no claims to wider generalizability.

**Procedural biases and measurement reliability**  Our goal in this study is to evaluate the ISBST system from the perspective of interactivity and to assess the applicability of SBST in an industrial setting. The former is related to the preferences of the engineers participating in the evaluation, since the effectiveness of the interaction is quite a personal issue. The latter, the applicability of the ISBST system in an industrial setting, is also dependent on the willingness of engineers to use it and accept it as part of everyday work. In such a context the measurements are, by necessity, subjective.

To ensure the reliability of the evaluation procedure, all participants were given full information about the nature, duration and purpose of the study. All participants received the same amount of training, information and time to evaluate the system. Every effort was made to isolate participants from distractions during the conduct of the evaluation.

Given that the purpose of the study was to evaluate the ISBST system, and that the participants were aware of this purpose, we do not deem evaluator
apprehension to be a significant threat. Moreover, observations taken during the evaluation revealed no sign of apprehension from any of the participants.

Several types of observations were recorded: The participants’ opinions, observations regarding their use of the ISBST system, and a set of system logs from their work with the ISBST system. Since no major disagreements were identified between these information sources, we conclude that the data collected can be trusted to be accurate.

Furthermore, since the participants are not familiar with SBST beyond a general level and the evaluation itself was largely exploratory, we deem it unlikely that there was any attempt to guess the ‘intended result’ that would endanger our findings.

For the experimental part of our study, the measurements were fully automated. As a result of this, we can conclude that those measurements are reliable.

**Random variations**  Random variations can occur in the participant population, and this can threaten the results of a study of this nature. However, all participants are engineers working in the same context and for the same company. Differences in skill and training may exist, but they are not expected to be of great enough magnitude to derail the findings of this evaluation.

**Researcher expectations**  This is a considerable validity threat, and addressing it was an important part in the evaluation design. Any questions that participants had were only answered before the evaluation itself began, with any assistance or advice limited to answering questions and providing a brief demo of the ISBST system. No participants asked for any further clarification doing the evaluation phase. Had there been any questions, the question itself, the answer provided, and their perceived effect on the participant, would have all been recorded and included in the analysis.

### 5.9 Conclusions and Future Work

In this paper we have presented an Interactive Search-Based Software Testing (ISBST) system, and have evaluated its applicability in an industrial setting. The system itself is designed to interact with domain specialists, in order to fully benefit both from their experience and from the power of search-based software testing methods. Simple, intuitive interaction with domain specialists is a key
factor in industrial applicability, since it makes such a system more usable and more easily accepted in an industrial setting.

An industrial evaluation was conducted and showed that ISBST is a useful addition in the context of our industrial partner, and complements existing testing methods. In addition, a follow-up experiment has shown that, while test cases can be developed from a static fitness function, user interaction is essential in developing interesting and useful test cases.

Overall, an ISBST system can be used by a domain specialist unfamiliar with search-based techniques to help test an embedded software module, while requiring a minimal effort in terms of training and generating test cases that complement existing approaches.

Future work will, of course, include further efforts to improve the Inner Cycle, without diminishing its capacity for generating the diverse potential solutions. Since we envision the system being used as an exploratory tool, to investigate those areas of a vast input space that a human would not think to test, ensuring that the back end provides diverse enough candidates is essential.

Clear avenues for further improvement of the ISBST system have also been identified. One example of such a potential improvement is extending the ISBST system to other modules, other companies and other contexts. The ISBST system was designed to be modular, specifically to allow easy extension to other modules within the same company and context.

Adapting the ISBST system to other companies in the same domain would be more involved. Subtle differences between companies mean that existing quality objectives would have to be validated and new objectives developed to match the new context. Changes and additional validation would also be needed to ensure that the information displayed is relevant in the new context.

Therefore, the improvement goal is to automate the integration of new SUTs in the same context, and to provide a clear roadmap for attempts to reuse the ISBST system in a new context, detailing. Changes of context could also be made easier by developing an editor that would allow the creation of new representations, new quality objectives, and that would synchronize the display mechanisms to the other components.

### 5.10 Acknowledgements

This research has been supported by funding from The Knowledge Foundation (KKS) in the project Next Generation Software Engineering (NGSE), project
5.10 Acknowledgements

no. 2010/0124. It was also partly funded by the KKS project no. 20130085 Testing of Critical System Characteristics (TOCSYC).
Chapter 6

Tester Interactivity makes a Difference in Search-Based Software Testing: A Controlled Experiment

Bogdan Marculescu, Simon Poulding, Robert Feldt, Kai Petersen, Richard Torkar

6.1 Introduction

Software testing plays a crucial role in increasing the quality of software systems, as well as the perceived quality of and confidence in such systems. One software testing technique is the application of metaheuristic optimization algorithms to generate test data, known as Search-Based Software Testing (SBST) [8, 9].

In a previous study [60], we have proposed a system that would allow successful application of SBST in an industrial context. This system, called the Interactive Search-Based Software Testing (ISBST) tool, facilitated the use of domain knowledge existing in the company to improve the search process. This was achieved by allowing human testers to interact with the system and guide the evolution of the search-based solutions. The interaction was inspired by work in Interactive Evolutionary Computation [10, 12, 27, 28, 36], and was de-
signed to allow the testers to make their contribution, without having to deal with the complexity of the underlying SBST system.

Previous work [73] focused on successfully applying ISBST in an industrial context, and determining what were the important factors that enabled successful application. One of the findings of that study was that the ISBST tool developed test cases that were quite different from those obtained by means of manual techniques. However, the evaluation was conducted with a low number of participants and in a context specific to our industrial partner, thus making it difficult to draw conclusions about the ISBST.

This study validates those findings, by conducting a large, controlled experiment, comparing the test cases developed using the ISBST system with those developed using a manual black-box technique. The experiment was conducted with 58 software engineering students, participants in a software Verification and Validation course at the master level. By selecting a more general SUT, in this case a clustering algorithm that is not tied to a particular problem domain or company, we can increase the level of confidence in the generalizability of our method. Master’s students are a good choice of participant, as they are not likely to be influenced by the biases and assumptions inherent in any domain, and more will have the time and willingness to participate in an experiment.

The experiment provides evidence that the automated system, represented by the ISBST tool, develops different test cases from the manual method. A follow-up computer-based experiment also provides evidence for the role of interaction in obtaining the results. By isolating the interaction strategy and comparing against the same search-based system without the benefit of interaction, we were able to provide evidence that interaction plays a significant role in the results obtained by the ISBST tool.

The contributions of this paper are as follows:

- Comparing test cases developed by the ISBST system and those developed by manual exploratory testing, to identify differences and similarities between them, and to determine whether or not they investigate the same type of SUT behavior.

- Assessing the effect of the interaction component of the ISBST system on the outcome of the search.

- Widening the application of the ISBST system to a completely new type of System under Test (SUT), part of a different domain.

- Evaluating the ISBST system on a wider set of participants, in a controlled environment.
Section 6.2 describes existing work on evolutionary approaches and search-based software testing and discusses the context of the current approach, as well as providing a description of the ISBST system itself. In Section 6.3 we describe the design of the current experiment and the tools used during the empirical process. Sections 6.4 and 6.5 present the results from the experiment and discuss their significance, respectively. The threats to the validity of the study are discussed in Section 6.6, and Section 6.7 concludes the paper.

6.2 Context

This experiment is inspired by results from a study conducted with our industrial partner, to investigate the possibility of using interactive search-based software testing (ISBST) to improve the testing process. Our industrial partner develops embedded software for industrial applications. The ISBST tool was previously developed and evaluated in that context, on a small number of company engineers. Therefore, this study will evaluate the ISBST tool outside of that specific context and with a larger number of participants.

We define a “domain specialist” as a person that develops and tests software for their specific domain as part of their activities, but that is not a software engineer. To assist domain specialists, tools are specifically designed to use the terminology, symbols, and concepts specific to the domain, rather than those specific to software development and testing. Thus, they focus on domain experience and expertise rather than knowledge specific to software testing.

In previous work [60] we proposed a tool, called the Interactive Search-Based Software Testing (ISBST) tool, that would use search-based techniques to help in the testing process. It is difficult to develop \textit{a priori} a fitness function that would be useful for a general SUT. As a result, the ISBST tool was designed to use a Dynamically Adapted Fitness Function (DAFF). In this concept, the fitness function is composed of a set of dimensions relevant to system quality to assess each candidate solution. By changing the relative importance of these attributes, the domain specialist can change the fitness function and indirectly guide the search. In our previous study, the relevant dimensions were identified and validated in collaboration with our industrial partner.

Further work [73] resulted in a practical implementation of the ISBST tool. The tool, and the concept of a Dynamically Adapted Fitness Function, were validated in a small case study conducted in an industrial setting. One of the results of that study was that the test cases that were developed by using the ISBST tool were useful and unexpected. The domain specialists using the tool
stated that they would not have considered investigating that type of behavior, but that the behavior itself was a good addition to the test suite.

The results of the exploratory study mentioned above indicated that using the ISBST tool would enable domain specialists to guide the search towards a more diverse set of behaviors than they could develop by using manual techniques. The more diverse set of behaviors would then be assessed by the domain specialists, who would refine relevant test cases and add them to the test suites.

Henceforth, we define the “behavior” as the set of measured outputs, or any function of those outputs, corresponding to a given set of inputs of the system under test (SUT). Thus, the “observed behavior space”, or just “behavior space”, is the total set of possible behaviors for a given SUT. Note that the behavior space deals only with characteristics of the SUT that are measured or evaluated, and is not a complete description of the SUT. The behaviors that are measured and form the behavior space will be called “behavior attributes”. The ISBST system may try to optimize, i.e. minimize or maximize, the found values for a given behavior attribute in a direction. In this case, a “search objective” is defined as the combination of behavior attribute and direction.

Additional behavior attributes may be identified and added, if they are considered relevant, and this would result in changes to the behavior space of the SUT. This further complicates attempts to explore the behavior space. For this paper, we define a “test case” to consist of a set of inputs and the corresponding SUT behavior.

The behavior space of a system is, in general, difficult to define and difficult to explore purposefully. Varying only certain characteristics of the behavior is, for most systems, a complex problem. The ISBST tool aims to use system behavior to measure the fitness of a given test case. By doing so, the ISBST tool can explore the behavior space of a system indirectly and develop test cases that explore previously unexercised, and unknown, regions of the behavior space.

6.2.1 Related Work

Search-based software testing (SBST) is the application of metaheuristic optimization methods to the problem of software testing. SBST is part of the larger scope of search-based software engineering, a term coined by Harman and Jones [7]. SBST has been successfully applied on a wide range of software testing problems. McMinn [9] describes the use of SBST for temporal, structural, and functional testing, while Afzal et al. [8] focus their review on the use of SBST on non-functional testing.
Search-based techniques, both in the wider area of software engineering and, more specifically in the field of testing, rely on having an automated means of assessing the quality, or “fitness” of a candidate solution.

However, the definition and understanding of what fitness is, and what candidates are preferable, can change during the search. This can be the result of a changing understanding of the problem, i.e. previously unknown information becomes available, or through clarifying misunderstandings or omissions, e.g. implicit domain knowledge not mentioned previously is now explicitly included in the fitness evaluation. As a result, designing a relevant fitness function \textit{a priori}, i.e. at the beginning of the process, has proven to be a challenge.

One way of addressing this issue was to engage human users in the search, to add their knowledge and intuition to the search. The user can interact with a search-based system at several levels of abstraction. At the highest level, the user sets the target that the search should reach, and allows the automated system the freedom to find solutions. At a medium level of abstraction, the human can change the way the fitness of candidates is being evaluated. The lowest level of abstraction puts the user in a position to directly influence how the search is performed, e.g. how new candidate solutions are developed.

An example of the highest level of abstraction would be an automated system that develops tests as the human user writes code or defines specifications [27]. The system would be influenced by the user indirectly, by having to adjust to the constantly changing goal. Indirect interaction could also be used to explore alternative designs, to understand design constraints or to assess alternative design decisions [12, 59].

At the medium level, a user can more directly guide the search by replacing the fitness function. Takagi proposed Interactive Evolutionary Computation (IEC), which he describes as an Evolutionary Computation (EC) “that optimizes systems based on subjective human evaluation” [10]. This would allow the human user to guide the search according to their “preference, intuition, emotion and psychological aspects” [10]. IEC could then see a wider spectrum of applications, including arts and animation.

Alternatively, a system may require the human user to only replace the fitness functions at certain times, e.g. to serve as a tie-breaker, when the existing fitness functions cannot rank certain candidates [64].

The fitness function itself can be subject to change, including user preference as a factor in computing fitness [68, 69], having elegance as a key factor in software design [70], or readjusting the fitness function to ensure that user preferred candidates receive a higher fitness score [67].
At the lowest level, interaction can be very detailed. Bush and Sayama [36] require the human to be “the main driver of the search process” by selecting the individuals and the evolutionary operators to be applied.

Replacing the fitness function with a human user, however, makes the fitness evaluation subjective and dependent on the individual user. This is not a problem for applications where subjective impressions are key, such as art, but might raise concerns when applied to engineering problems.

A more serious problem is that the number of evaluations that a human can perform is limited, as boredom and fatigue will set in. This is even more of an issue at the lowest level of abstraction, where the human user is involved in evolving each candidate. Fatigue has already been identified as a major concern, and efforts to alleviate the problem have been proposed [61]. Alternatives have, therefore, been proposed that make it easier for the human user to interact, by selecting candidate solutions they favor and dismissing those they do not [37]; or focusing on the search objectives or the fitness values more than on the candidate solutions themselves [38, 39].

Existing work on interaction in evolutionary computation seems to be focused on areas other than software testing. Nevertheless, the interaction techniques being described are applicable on any search-based system, as long as elements of it are subject to human evaluation. Moreover, human preference can help guide the process where the objectives of the search are unknown or unknowable. This is evident in applications such as aesthetics and software design, but ambiguity exists in other areas as well.

6.2.2 Interactive Search-Based Software Testing (ISBST)

The ISBST tool is designed to make it easy for a domain specialist to support the search process with their knowledge and experience, without requiring familiarity with the particular implementation of the underlying algorithm. In terms of the levels being described above, the ISBST system exists at the higher level. Users of the system will interact with the system to develop the fitness function and provide an evaluation of some of the resulting test cases, but not replace the fitness function or evaluate each individual test case. This approach is aimed at allowing the user to control how the search proceeds for the entire population, rather than focusing in on individual candidate solutions.

The ISBST system generates, based on guidance from the domain specialist, a population of candidate solutions or “candidates”. An overview of this population of candidates is provided to the domain specialist. The domain specialist can select from the population candidates that are of interest, obtain more in-
6.2 Context

The system is composed of two nested components, the inner cycle that contains all the components for initiating, running, and guiding the search, and the outer cycle that handles the interaction with the user, as shown in Figure 6.1. The outer cycle interacts with the user periodically, displays the candidates, collects the inputs, and then resumes the search with the new input. Between two interactions, the inner cycle is run with the selected inputs.

The Inner Cycle consists of the search algorithm itself, the means of interacting with the SUT, and the means for guiding the search. When the search algorithm generates a new test case candidate, that candidate consists only of a set of inputs. The inputs are then fed into the SUT, to obtain the corresponding behavior. Once the candidate is complete, its fitness is evaluated by means of a fitness function.

Defining a fitness function a priori is a difficult task, as it is impossible to guess the specific details of each system and each situation. To handle search guidance, the ISBST system uses a number of behavior attributes. The behav-
ior attributes are generally SUT specific or, at most, domain specific. In our experience so far, behavior attributes have been defined during the development process, relying on input from domain specialists, and validated with the help of domain specialists. Each behavior attribute has a weight, set by the domain specialist via the outer cycle. The weight represents the importance of each behavior attribute with respect to the others.

The fitness of a candidate is a weighed sum of the scores obtained by that candidate for each of the behavior attributes:

$$\text{DAFF}_j = \sum_{i=1}^{n\text{Objectives}} \text{Weight}_i \cdot \text{Value}_{i,j}$$  \hspace{1cm} (6.1)

where $\text{DAFF}_j$ is the fitness value of candidate $j$, $\text{Weight}_i$ is the current weight of the objective $i$, and $\text{Value}_{i,j}$ is the fitness value of candidate $j$ measured by objective $i$. The value of $\text{DAFF}_j$ is the sum of the weighted fitness values for all $n\text{Objectives}$ objectives. An objective $k$ can be deselected from the computation by having $\text{Weight}_k = 0$.

The relative importance of each behavior attribute is the mechanism by which the domain specialists can influence the search: values for the weight of each objective are set by the domain specialists in the outer cycle, and then passed to the inner cycle and used in the fitness evaluation.

**The Outer Cycle** enables the domain specialists to interact with the ISBST system. This interaction has two components: visualizing candidate solutions and guiding the search.

The domain specialist is shown a summary of the current population of candidates solutions and overview of the scores obtained by the candidates for each behavior attribute. Each candidate’s scores for each behavior attribute, and detailed information regarding each candidate are available on demand. This information is domain and even SUT specific.

The user guides the search by setting the importance of each behavior attribute with respect to the others. This is done by assigning a weight to each behavior attribute. The weight is passed to the inner cycle and used to compute the DAFF, and therefore guide the search.

This guidance is achieved by allowing the domain specialist to adjust the relative importance of each of the search objectives. The objective, with their assigned weights, then form the DAFF that is used for the next set of optimization steps. After a number of optimization steps, the domain specialist is shown the latest generation of test case candidates, provided with all the information available on those candidates, and offered the possibility to adjust their
weighting accordingly. This interaction with the specialist is called “interaction event”.

The exact purpose of each test case is determined by the domain specialist, as is its fitness for that purpose. The ISBST tool generates the input data for the test cases, computes the SUT behavior corresponding to those inputs, and then evaluates the fitness of the test case according to the weighting provided by the user. As the weighting changes, so does the optimization objective of the ISBST tool.

The ISBST tool does not require the availability of an oracle to specify if the observed behavior is acceptable or not. Since the goal is to investigate areas of the input and behavior space that human domain specialists would not have otherwise considered, it can be expected that other mechanisms to assess that behavior, e.g. specifications, models, etc., are not available. The expectation is that domain specialists would identify those of the generated test cases that are remarkable in some way and assess that behavior themselves.

Once a candidate, or group of candidates, has been found to be suitable, they can be exported for later review and for inclusion in test suites. The search process can then resume.

6.2.3 The ISBST Tool Implementation

The system is implemented as a distributed system, with the inner cycle being implemented in the Julia language\(^1\) and deployed on a remote server. The outer cycle is implemented as a browser based application in Javascript, using the Data-Driven Document (D3) library\(^2\) to enable an informative candidate display. Communication between the two sides is done by packaging candidate objects in a JavaScript Object Notation (JSON) file\(^3\). The system is run as a server application, and experimental participants need only connect to it via browser.

During an interaction event, the user connects to the outer cycle via the web page. There they can see an overview of the candidate solutions, plotted according to the scores they obtained for each quality objective, as shown in the example in Figure 6.2. Detailed information for each candidate solution is also available on demand, for example in Figure 6.3, to reduce the risk of overwhelming the user with information. To guide the search, the users set the

---

\(^1\)http://julialang.org/
\(^2\)http://d3js.org/
\(^3\)http://www.json.org/
(a) Overview of the test space.

(b) Detailed view of the test space.

Figure 6.2: Example of an overview of the solutions found by the ISBST system (current generation in blue, previous generation in orange). On the left, an overview of the candidate solutions plotted with respect to all the behavior attributes. On the right, a detailed view of one of the subgraphs. The graph shows an example of a candidate population, plotted according to two of the relevant behavior attributes (weight_range and mean_silhouette, in this case). The matrix of scatterplots can be extended to reflect an arbitrary number of behavior attributes.

Figure 6.3: Detailed view of one candidate solution. The detailed view shows the score obtained by the candidate for each of the behavior attributes. This panel shows the raw values for each score, rather than the normalized values used in the calculations.
6.2 Context

Figure 6.4: Detailed view of the search guidance panel. All the behavior attributes are specified here, and the domain specialist can view and alter the weight of each attribute to reflect their relative importance at any given moment.

relative importance (weight) of each behavior attribute by means of a set of sliders. An example of that panel of sliders can be seen in Figure 6.4.

When the user clicks the “Resume Search” button, the search resumes until the next interaction event. In the current implementation, the interval between interaction events is fixed: after 50 optimization steps, a new interaction event is triggered.

For the current version of the system, the search algorithm used is a Differential Evolution algorithm [49]. Differential Evolution is a parallel direct search method, where each potential solution is encoded as a vector of real numbers. In our case, each vector represents the input data in one test case. The initial population is chosen randomly from a uniform distribution, and covers the entire parameter space. New parameter vectors are created by mutation: adding the weighted difference between two population vectors to a third vector.

For each target vector \( x_{i,G} \), where \( i = 1, 2, \ldots, NP \) a mutant vector is generated as follows:

\[
v_{i,G+1} = x_{r_1,G} + F \ast (x_{r_2,G} - x_{r_3,G}) \tag{6.2}
\]

where \( r_1, r_2, r_3 \in 1, 2, \ldots, NP \), are integers, and mutually different, and different from the running index \( i \). \( F \) is a real and constant factor \( \in (0, 2] \) which controls the amplification of the differential variation \( (x_{r_2,G} - x_{r_3,G}) \). If the mutant vector is an improvement over the target vector, it replaces it in the following generation [49].
The crossover rate we used is \( cr = 0.5 \), the scale factor is \( F = 0.7 \), and the population size is 100. The mutation strategy is that proposed by Storn and Price [49]; \( DE/rand/1/bin \). The strategy uses a differential evolution algorithm (DE); the vector to be mutated is randomly chosen (rand); one difference vector is used (1); the crossover scheme is binomial (bin).

To allow the single objective DE to handle multi-objective and many-objective problems, we used the Sum of Weighted Global Averages [50]. Individuals that have higher fitness scores in more important search objectives, i.e. that have a higher weight, will receive a better overall fitness score. This ensures that the search favors those individuals that the users, by means of the weights, have decided are important.

The search objectives may represent a large number of different measurements and have very different scales. The fitness value for each search objective is normalized based on the extreme values for that objective, as shown in Equation 6.3. This ensures that a search objective cannot influence the fitness function by virtue of its scale, rather than its relative importance as assessed by the user.

The fitness ratio of candidate \( i \) for one behavior attribute is shown below, in Equation 6.3.

\[
\text{fitness} \_\text{ratio}_i = \frac{(\text{value}_i - \text{min}_i)}{(\text{max}_i - \text{min}_i)}
\]  

(6.3)

where \( \text{fitness} \_\text{ratio}_i \) is the fitness ratio of candidate \( i \) with respect to one behavior attribute, \( \text{value}_i \) is the raw score obtained by candidate \( i \) with respect to the current behavior attribute, and \( \text{min}_i \) and \( \text{max}_i \) are the globally best and worst values seen for the current objective in the entire search.

### 6.3 Experimental Design

The purpose of the experiment is to evaluate the ISBST system in a controlled laboratory setting. We hypothesize that the ISBST system finds test cases that are not developed by human users alone, and therefore testing results in different behaviors of the SUT.

The experiment was designed to answer the following research questions:

- **RQ1.** Do test cases developed by the ISBST system investigate different regions of the behavior space? If so, is the difference significant?
The main hypothesis of the experiment is that test cases developed using the ISBST system are different in terms of the SUT behavior they cover from test cases obtained using the manual black-box testing technique. By investigating different regions of the behavior space, the ISBST system would increase the diversity of available test case candidates.

While we acknowledge that test case diversity does not necessarily imply increased fault finding ability, this remains a reasonable objective. Work by Feldt et al. links test suites with higher diversity to higher structural and fault coverage [74, 75].

• **RQ2.** Do both the search and the interaction components of the ISBST system have a significant contribution to the observed effects?

Our hypothesis is that both components of the ISBST system have a contribution to the result and, thus, both are relevant. To answer this question, we executed a second experiment, to evaluated the impact of the search and interaction components in isolation. To achieve this goal we ran the ISBST system again, with the same settings, but without the interactive components.

• **RQ3.** Is the ISBST system more demanding of the domain specialist than using the manual exploratory testing technique? We wanted to investigate in greater detail the demands that the ISBST tool places on the domain specialist. The ISBST tool adds another layer of abstraction, so we hypothesize that it will also place a greater strain on the domain specialist.

The independent variable is the method being used: manual or ISBST. Both methods are supported by tools that make the same information available to the participant, and computed in the same way. Thus, any difference being observed, will not be due to data availability, or differences in the algorithms for behavior computation.

The dependent variables are: the set of test cases produced by each of the methods in the two experiments and the auto-assessed level of demands placed on the participants.

### 6.3.1 Participants

The participants to the experiment were 58 students from a software engineering Master’s program at the Blekinge Institute of Technology, in Sweden. All
the participants were students in the Verification and Validation course, and were recruited for the experiment through the course. All the participants were volunteers. The incentive provided during the recruitment process was the opportunity to use some of the techniques taught during the course in a more practical setting. No other rewards, financial or course credits, were offered to the participants.

The Verification and Validation course aims to teach students the importance of systematic verification and validation of software, and provides them with knowledge of available methods and techniques, complete with their capabilities and limitations. The course covers methods such as reviews, unit testing, coverage, statistical approaches, system and integration testing, reliability, and performance. During the course, students are encouraged to critically reflect and discuss topics in verification and validation, as well as to critically evaluate the strengths and weaknesses of verification and validation techniques. In addition, students gain practical experience in planning and applying test strategies and tools on open source systems and conducted automated source code inspections.

The experiment was conducted at the end of the Verification and Validation course, so participants benefited from the knowledge obtained during that course, in particular from the practical exercises. That being said, no additional guidance was provided by the experimenter, and the way participants approached the task was deliberately left up to individual decision.

The results of the initial participant characterization survey show most participants have some expertise in software development, mostly at a theoretical level, due to the courses taken during their education. Most had no industrial experience in software development or testing. Most participants also believed they had a theoretical knowledge of the domain and some knowledge of statistics, due to courses taken during their education, but had not used this knowledge in practice. A basic introduction into SBSE was included in the course, so all participants had some basic knowledge of the technique.

We tried to isolate the method as the only variable that differed between the two groups of participants, to the extent to which this was possible. The relatively homogeneous level of knowledge and expertise lead us to conclude that any effects observed are due to differences between the ISBST and the manual tools, and therefore of the techniques, rather than differences in terms of experience between participants.

As students in the Verification and Validation course, the participants were familiar with the manual testing technique and had been introduced to ISBST. Thus they were motivated to participate and had the capabilities to complete the tasks.
6.3 Experimental Design

6.3.2 System under Test

The system under test (SUT) chosen for this experiment is the Julia language\(^4\) implementation of a \(k\)-means clustering algorithm. The implementation is available as a Julia library\(^5\), complete with documentation\(^6\). The library consists of 5438 lines of code, in 58 functions, written entirely in Julia.

This system was chosen for two reasons. The first reason is the level of complexity. We define complexity both quantitatively - as the number of inputs and outputs that form each potential solution, and qualitatively - as the difficulty a participant may face in evaluating proposed solutions.

The chosen system had to be simple enough to allow for exploratory testing within the limited time available. A more complex system than the one chosen would have a detrimental impact on the performance of the manual method and make comparison more difficult. At the same time, the system had to be complex enough to be representative of the embedded software that was the inspiration for the experiment.

The second reason has to do with assessing the outcome of the SUT. For the embedded systems that served as inspiration for this study, a human expert is considered to have the most competence and experience to evaluate a proposed solution.

For the clustering SUT chosen for this experiment, we have quantitative measurements available to assess the proposed solutions as well as the human participants providing a qualitative “sanity check”. This combination allows us to assess both methods, both in terms of the human participants’ perception of solution quality and in terms of the quantitative evaluation of the different objectives. For each test case candidate, the testers can see the scores for each of the quantitative measurements and can see a graph of the clustering results. Based on that information, they decide if that is an interesting enough test case to add to the test suite or not.

The behavior of the SUT, for this experiment, consists of the behavior attributes shown in Table 6.1. The behavior attributes are based on measurements developed to assess and validate clustering results. The aim, whether the behavior attribute is to be minimized or maximized, was arbitrarily chosen for the experiment.

In practical terms, for each test case, the set of inputs consists of 60 points in a two-dimensional space, and the desired number of clusters. An example of

\(^{4}\)http://julialang.org/
\(^{5}\)https://github.com/JuliaStats/Clustering.jl
\(^{6}\)http://clusteringjl.readthedocs.org/en/latest/
<table>
<thead>
<tr>
<th>No.</th>
<th>Behavior Dimension</th>
<th>Description</th>
<th>Direction</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of Clusters</td>
<td>Number of clusters to be found. (The ( k )-means algorithm required the number of clusters to be found as an input)</td>
<td>Minimize</td>
</tr>
<tr>
<td>2</td>
<td>Number of Iterations</td>
<td>( k )-means clustering is done in several iterations, until the clusters are stable.</td>
<td>Minimize</td>
</tr>
<tr>
<td>3</td>
<td>Mean Silhouette</td>
<td>The Silhouette is a quantitative way to measure how well each item lies in its own cluster. The Silhouette of each point has a value between 0 and 1, with higher values (closer to 1) indicating that the point lies well within its own cluster and there is no meaningful alternative cluster it could be assigned to. The mean is computed to provide an overview of how well the points belong to their respective clusters across the entire population.</td>
<td>Maximize</td>
</tr>
<tr>
<td>4</td>
<td>Silhouette Range</td>
<td>This is the absolute distance between the lowest and the highest silhouette values found in the current candidate. A high value for this attribute means that the test case contains both well-defined and ill-defined clusters. A low value indicates that the test case contains only one of the two options.</td>
<td>Maximize</td>
</tr>
<tr>
<td>5</td>
<td>Mean Weight</td>
<td>The weight of a cluster is the sum of the weights of all the points within it. For the purpose of this experiment, each point has the same weight: ( \text{weight} = 1 ). Larger clusters, with more widely dispersed points, will get high values for this quality objective. Small, tightly packed, clusters will get low values for this objective.</td>
<td>Minimize</td>
</tr>
<tr>
<td>6</td>
<td>Weight Range</td>
<td>Test cases containing a combination of large and small clusters will obtain a high value with respect to this objective.</td>
<td>Maximize</td>
</tr>
</tbody>
</table>

Table 6.1: An overview of the Search Objectives used in this experiment.
the inputs can be seen in Figure 6.5. The SUT is then run with these inputs, and the behavior of the SUT is the set of values obtained for each of the fitness attributes.

6.3.3 Research Instruments

The Measurement Aid. The manual technique used is an exploratory black box testing approach, supported by a task specific tool.

Manually writing up the points required for a test case could get tiring and frustrating, and thus impact the evaluation. To ensure that participants using both methods have access to the same type of information about the test cases being developed, and to mitigate the risk of fatigue or frustration unduly affecting the results, a manual testing helper tool was also developed.

Participants to the experiment are expected to manually develop the inputs to the test case, i.e. the 60 points to be clustered and the number of clusters. The SUT is then run with the selected inputs and the behavior of the SUT is shown. The tester decides the inputs, and then can see the resulting scores for each behavior attribute and the graph of cluster assignments. Based on that information, they can decide whether to save the test case or not.

The tool allows for an easier way to draw up the test case, to view the behavior of the SUT, and to save interesting test cases, using the panel shown in Figure 6.5. Participants are provided with a more convenient and intuitive way to arrange the inputs for the clustering algorithm. The Measurement Aid randomly generated the required points. Then, the participant can drag any of the points in the two-dimensional input space, to match the desired set of inputs. The coordinates of the point are then updated. Once all the points are where the participant requires them to be, the inputs are sent to the SUT, the behavior is computed, and the test case is displayed to the participant.

As mentioned previously, the behavior system consists of more than the cluster assignment, but also includes various means of assessing and describing the result. The output information presented to participants using the manual tool is the same as the information presented to those using the automated tool, and displayed in the same manner. This ensures the two methods are comparable, and that neither method has access to more information that the other.

NASA Task Load Index. The NASA Task Load Index (NASA-TLX) is a subjective workload assessment tool described by Hart and Staveland [76]. The NASA-TLX is commonly used to subjectively assess the workload when working
Figure 6.5: Defining the inputs for the manual tool. The points can be moved individually to desired positions. In the current figure, the points are color coded according to the cluster they were assigned to.
with human-machine systems [77] in general and on assessing task difficulty in
software development [78] in particular.

Participants were provided with a paper and pen version of the NASA-
TLX evaluation, to be filled in twice by each participant: after each of the two
experimental sessions. The most important dimensions for the purpose of this
experiment were: mental demand, effort, temporal demand, frustration, and
performance.

The results for each dimension were analyzed separately, rather than being
unified into a single demand rating. This allowed us to gain a more detailed
understanding of how each method was perceived by the participants. It also
allowed us to avoid uncertainties relating to the relative importance of each of
the dimensions.

6.3.4 Experimental Process

The experiment is a crossover design, as shown in Figure 6.6. The experiment
consists of two treatments: using the ISBST system and the Manual technique,
noted in Figure 6.6 as “ISBST” and “Manual”. The participants are randomly
split into two groups, and the experiment consists of two sessions. One group,
Group 1 in Figure 6.6, will use the ISBST system for the first session and the
Manual method for the second, with the second group doing the reverse. This
ensures that all participants have a chance to use both techniques.

The design of the experiment was refined and validated with a pilot experi-
ment, conducted with other researchers. The pilot was full length and allowed
experimental procedures, documents, and the timing of the process to be re-
fined. The pilot was used to evaluate the procedure, the clarity and quality of
the instructions given to the participants, assess the duration of the experiment,
and identify any other practical considerations that might affect the experiment.
While the experiment followed the same procedure, the following details concern
the final experiment.

The experiment consists of two treatments, both aimed at developing a di-
verse set of test cases. The first technique is the ISBST system described in
Section 6.2.2 in detail in our previous work [60, 73].

For the second treatment, test cases are developed manually. Each partici-
pant selects the inputs for the SUT, i.e. the 60 points and the desired number of
clusters. The tool described in Section 6.3.3 then executes the SUT with those
inputs and returns the behavior of the SUT.
Due to limitations in the available laboratory space, the experiment was conducted in three separate experimental instances, all conducted within 10 days. The participants were assigned to the experimental instances randomly.

Each experimental instance began with a 20-minute presentation and demo. The presentation provided an initial description of the experiment, of the tools that would be used, and of the system under test. It also included a detailed guide describing how the participants can develop test cases using each tool and explaining the data each tool provides. The demo showed participants how to use the two tools and what they should expect from them. The presentation also emphasized the goal of the participants: to create a regression test suite that covers a wide range of interesting behaviors. The emphasis on covering many interesting behaviors is necessary, since the goal is to evaluate areas of the behavior space that each of the method covers.

The experimental instance consisted of two 45-minute sessions. During a session, a participant would use the technique assigned to them to develop test cases. Test cases that were considered useful or interesting by the participants were then saved.

All participants used both of the methods, with the order in which each participant used the methods being randomly assigned. Each session concluded with completing an evaluation form based on the NASA-TLX. The two experimental sessions were separated by a 15-minute break.

In addition, each participant had a few additional documents available. First, a document providing a brief description of the behavior attributes that constitute the SUT output, an explanation of the practical consequences of high or low values for each attribute, and what the goal that the automated method was trying to reach for each attribute. A second document provided information on the method each participant used, information on how to interact with each tool, and other practical information needed to successfully complete the experiment.
6.3 Experimental Design

Since there was a break of a few days between experimental instances, as well as a short break between experimental sessions, we have asked participants not to discuss the details of the tasks, or the approach they used, with their colleagues. We have also made it clear to the participants that their performance on the experiment would not be used as an assessment for the course and would not impact their grades, to reduce the incentive participants had to obtain additional information from their colleagues. It was also made clear to the participants that there is no “right” or “wrong” answer that they are meant to find.

The experimental sessions were conducted under observation, with the same researchers being involved in all sessions. This was done to ensure that all experimental sessions received the same instructions, presentation, and information. Participants have the opportunity to ask questions, and care was taken that clarifications were not leading the participants to expected behaviors.

6.3.5 Data Collection and Analysis

RQ1 is concerned with assessing the degree to which test cases developed by the ISBST system differ, in terms of the SUT behavior they explore, from test cases developed using the manual technique. The null hypothesis in this case is that there is no significant difference between the regions of the behavior space that each method explores.

To answer RQ1, we collected the test cases developed by the participants during the experimental sessions and specifically marked for “export” or saving. This allows participants to only include test cases they regard as interesting or novel enough to consider. Since interaction with the domain expert is one of the key attributes of the ISBST system [60], in answering RQ1, we only considered test cases that participants had decided to select.

To understand the complex and extensive data that we collected, we have used several analysis techniques. First, we evaluated whether there was a statistically significant difference between the ISBST and the manually developed test cases, in terms of the regions of the SUT behavior space they investigate. To determine if the results were statistically significant we used a non-parametric test: the Mann-Whitney U-test [79], as recommended by Arcuri and Briand [80]. Effect sizes were calculated and interpreted using the method proposed by Vargha and Delaney [81].

In addition, we wanted to look at which areas of the behavior space were explored by each method. To achieve this, we clustered the test cases based on their behavior, and analyzed the composition of the clusters. If a cluster
contains test cases resulting from one method only, we can conclude that that area of the behavior space is only explored by one of the methods.

Principal Component Analysis (PCA) [82] was used to isolate the behavior attributes that accounted for most of the variation. This type of analysis would highlight differences in the regions of the behavior space covered by each experimental treatment and could be used to confirm that any results are caused by the different characteristics of the methods, and to strengthen confidence in the other analysis methods. The dimensionality reduction provided by the PCA also allows visualization of the two groups of test cases so that any overlap, or lack of an overlap, can be more directly judged.

To answer RQ2 we had to isolate the search-based system from the interactive component and assess their performance individually. A laboratory experiment was conducted to evaluate the effect of the search and the interaction components separately.

To assess the performance of the interaction component, test cases developed by the participants using ISBST system were compared to those developed by the ISBST system without the benefit of interaction. The ISBST system without interaction assumed that all objectives have the same importance and assigned all the objectives the same relative weight.

To make sure that such a comparison would not be influenced by any biases in the test cases participants exported, we collected the entire population of test cases developed by the ISBST system, independent of whether they were selected for export or not. The Mann-Whitney U test was used for the comparison, and effect sizes were calculated and interpreted using Vargha-Delaney.

We also collected information regarding the interaction between each participant and the ISBST tool: the number of interactions and the weights each participant used for the objectives in each interaction.

The realistic interaction data consists of the populations of test cases automatically collected at the end of the experiment. For each participant, the final population of 50 test cases was recorded, as was the number of interaction events each participant used.

For each participant, a separate search was started, and the system was run for the same amount of interaction events as the participant had used in the practical experiment, and using the same settings. The only difference was that the interaction strategy each participant used was replaced with a Null Strategy. The Null Strategy consists of keeping every objective weight to the same, non-zero, weight. This results in all the objectives having equal priority, and is equivalent to a search conducted with no interaction. We will refer to
the experimental runs using the Null Strategy as “non-interactive executions” of the ISBST system.

To allow us to better determine whether or not interaction has an impact on the outcome, both the practical experiment and the non-interactive execution had the same number of optimization steps available, and thus the same number of fitness evaluations. This is to clarify that the goal is not to assert the dominance of one approach over the other, but to determine whether the interaction component has an impact on the outcome: i.e. that the search does not converge to the same results regardless of the human interaction. This method of assessment, recommended by Črepinšek et al. [83], allows for a fair comparison between search algorithms.

To answer RQ3, as well as to get an overview of the participants to the study, information was also collected regarding their experience, skills, strategies, performance, and the level of fatigue incurred by using the methods.

Descriptive statistics, conducted on the participant data, provided an understanding of the participants’ level of expertise, experience, their strategies, performance and level of demand. This approach provided insight into the degree to which the two methods are comparable in terms of the expertise and effort required.

### 6.4 Results

#### 6.4.1 Test Suite Comparison

To answer RQ1, we looked at the test cases developed by the participants. During the course of the experiment, the 58 participants developed a total of \( n_{\text{total}} = 4615 \) test cases, of which \( n_{\text{auto}} = 4154 \) were developed and exported using the ISBST tool and \( n_{\text{manual}} = 461 \) were developed using the manual tool. This imbalance in terms of numbers is expected, as the ISBST tool generates a larger number of test cases in the same period of time than the manual method. The analysis focuses on the behavior of the test cases. We defined the behavior of a test case as the set of scores obtained by that test case with respect to the objectives described in Table 6.1.

A first look at the data consisted of performing the Mann-Whitney U test on each dimension of the test case behaviors. The Mann-Whitney U-test is a non-parametric test, so no assumptions need to be met about the distribution of the population. The purpose of this analysis was to determine if there is a difference between test cases developed using the ISBST tool and the man-
Figure 6.7: The difference between test case behavior (on each behavior dimension) between test case populations developed by the manual technique and the ISBST system.

ualy developed test with respect to each objective that describes the output. This initial analysis shows a statistically significant difference between the data developed by the two methods, with $p$-values of $< 10^{-5}$. The values for each objective can be found in Figure 6.7, and the results of the statistical analysis in Table 6.2. Thus, we can state that the test cases resulting from the two methods differ from each other with respect to all the objectives that define the output.

To get a better understanding of how the test cases are distributed through the behavior space, we performed a hierarchical clustering on the objective scores. The scores were clustered using Ward’s minimum variance method [84] and employed the Euclidean distance as a metric. By clustering the test cases based on their behavior, we wanted to obtain distinct areas of the behavior space that were composed of similar test cases. We could then assess if there were any such areas that only had test cases resulting from one method. This would indicate that that region of the behavior space was explored by one method, but not the other. A cluster that contained test cases from both methods would indicate a region of the behavior space that both methods had explored. An overview of the results of the hierarchical clustering can be seen in Figure 6.8.
6.4 Results

<table>
<thead>
<tr>
<th>Effect Size</th>
<th>Number of Clusters</th>
<th>Number of Iterations</th>
<th>Mean Silhouette</th>
<th>Silhouette Range</th>
<th>Mean Weight</th>
<th>Weights Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>Interpretation</td>
<td>negligible</td>
<td>medium</td>
<td>medium</td>
<td>large</td>
<td>medium</td>
<td>small</td>
</tr>
<tr>
<td>p-value</td>
<td>$&lt;10^{-5}$</td>
<td>$&lt;10^{-5}$</td>
<td>$&lt;10^{-5}$</td>
<td>$&lt;10^{-5}$</td>
<td>$&lt;10^{-5}$</td>
<td>$&lt;10^{-5}$</td>
</tr>
</tbody>
</table>

Table 6.2: The $p$-values of the Mann-Whitney U test, for the differences (assessed in terms of behavior) between tests developed by the ISBST tool and those developed by the manual method. The effect sizes were calculated and interpreted using the Vagha-Delaney A measure.

![Hierarchical clustering of test case behaviour](image)

Figure 6.8: An overview of the hierarchical clustering. The y-axis is a measure of closeness of the clusters. Each leaf on the x-axis is a test case. They are arranged in order, based on the cluster assignment. The large number of test cases made labeling individual test cases impractical. As a result, for each manually developed test case a line is drawn in red, on the bar at the bottom of the image. For the test cases developed using the ISBST system, a similar line is drawn in blue. The red boxes indicate the clusters used in the analysis. The width of each cluster is proportional to the number of test cases it contains. The left-most two clusters contain only manually developed test cases. The fourth and fifth cluster from the left only contain test cases developed by the ISBST system.
We cut off the hierarchical clustering after obtaining 6 clusters. The value was chosen arbitrarily, as it seemed to provide a finely grained view of the behavior space without increasing the complexity to unmanageable levels.

The result, shown in Figure 6.8, shows the three larger clusters on the right of the image composed mostly of test cases resulting from the ISBST tool, with two of them being composed exclusively of test cases obtained by that method. The two clusters on the left of the figure are composed solely of test cases obtained by manual exploratory testing.

Table 6.3 shows the distribution of the number of test cases for each cluster. It can be seen that, while some overlap does exist, a large number of the test cases are in clusters that have no overlap. This seems to indicate that the two methods are focusing on different areas of the behavior space.

These observations are also supported by the results of a principal component analysis, conducted on the behavior space. Three dimensions account for 76.74% of the observed variability. Since displaying three dimensions makes the graph harder to interpret, we have chosen to focus on the two most influential dimensions. These account for 62.57% of the observed variability and are shown in Figure 6.9.

The individual test case behaviors were plotted and form distinct, though occasionally overlapping, clusters in the behavior space. The manually developed test cases (seen in red in Figure 6.9) tend to be more spread out and cover a different area of the behavior space than those developed by the ISBST system (shown in blue in Figure 6.9). The optimization objective for the search-based algorithm was the upper right corner of the graph. Both the manual technique and the ISBST system use the same objectives, i.e. the same behavior dimensions with the same directions for optimization. Random test cases were also developed, for comparison purposes and are shown in Figure 6.9 in green.

Thus, we can state confidently that the two methods investigate different areas of the behavior space.

<table>
<thead>
<tr>
<th>No.</th>
<th>Number of ISBST-generated test cases</th>
<th>Number of manually generated test cases</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>502</td>
<td>0</td>
</tr>
<tr>
<td>2</td>
<td>1392</td>
<td>0</td>
</tr>
<tr>
<td>3</td>
<td>60</td>
<td>293</td>
</tr>
<tr>
<td>4</td>
<td>2200</td>
<td>35</td>
</tr>
<tr>
<td>5</td>
<td>0</td>
<td>75</td>
</tr>
<tr>
<td>6</td>
<td>0</td>
<td>58</td>
</tr>
</tbody>
</table>

Table 6.3: The distribution of test cases to the clusters.
Figure 6.9: The results of a principal component analysis conducted on the test case behavior space. The tests obtained from the manual method are marked in red, those obtained from the ISBST tool are in blue, and those generated randomly are in green. The upper right corner is where an “ideal” solution would have optimal values for all dimensions.
Figure 6.10: The behavior of the test cases with respect to Mean Silhouette and Mean Weight. The test cases obtained from the manual method are in red, those obtained from the ISBST tool are in blue. The origin of the graph (lower left corner) is where an ideal solution would have optimal values for all dimensions.

A closer analysis also shows differentiation in terms of the different objectives. Some of the objectives allowed the participants to find a front of solutions, illustrated in Figure 6.10 in red, even if that was dominated by the test cases developed by the ISBST system (shown in blue in the same figure). The dimensions in the figure were chosen because they best illustrate the solution front.

For other objectives, e.g. Silhouette Range and Weights Range, the front is a lot less clear, as can be seen in Figure 6.11. In this case, the manually developed test cases are more spread out in the behavior space and show less evidence of a front. The two dimensions in the figure were chosen because they more clearly illustrate this.

A key element in the application of the ISBST tool is the complete and correct definition of objectives. In previous studies [73] this was achieved by means of validating the objectives with domain specialists at the company. Such validation would, however, be impractical for a system aimed at a wider audience or when domain specialists are not available. Objectives that are more difficult to optimize, e.g. those in Figure 6.11, may hide improvements in other selected objectives.
6.4 Results

Figure 6.11: The behavior of the test cases with respect to Silhouette Range and Weights Range. The test cases obtained from the manual method are in red, those obtained from the ISBST tool are in blue. The top right corner of the graph is where an ideal solution would have optimal values for all dimensions.

As a result of this analysis, we can conclude that the answer to the first research question is that the two methods, the ISBST tool and the manual exploratory testing, investigate different areas of the behavior space. We have found that there are regions of the behavior space that are only explored by only one of the two methods.

6.4.2 The Effects of the Search and of Interaction

The ISBST tool is composed of two elements: the search component (identified in Section 6.2.2 as the Inner Cycle) and the interaction component (called Outer Cycle). The results up to this point have shown that the ISBST tool, taken as a whole, achieves the goal of investigating areas of the search space that the manual method does not reach.

The effect of the search component of the ISBST tool is the difference between the initial population at the beginning of the search, and the population at the end of the non-interactive execution.

Figure 6.12 shows the difference between the initial and final populations in the non-interactive execution. It is clear that for those objectives that were to be maximized (Weights Range, Mean Silhouette, and Silhouette range) the final population shows higher values than the initial one. Conversely, for the
<table>
<thead>
<tr>
<th>Number of Clusters</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>5</td>
<td>10</td>
</tr>
<tr>
<td>15</td>
<td>20</td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Weights Range</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.25</td>
<td>0.30</td>
<td>0.35</td>
</tr>
<tr>
<td>0.40</td>
<td>0.45</td>
<td>0.50</td>
</tr>
<tr>
<td>0.55</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Silhouette</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>6</td>
<td>8</td>
<td>10</td>
</tr>
<tr>
<td>12</td>
<td>14</td>
<td>16</td>
</tr>
<tr>
<td>18</td>
<td></td>
<td>20</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mean Weight</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>0.6</td>
<td>0.8</td>
<td>1.0</td>
</tr>
<tr>
<td>1.2</td>
<td>1.4</td>
<td>1.6</td>
</tr>
<tr>
<td>1.8</td>
<td></td>
<td>2.0</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Silhouette Range</th>
<th>Initial</th>
<th>Final</th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>15</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Number of Iterations</th>
</tr>
</thead>
<tbody>
<tr>
<td>3</td>
</tr>
<tr>
<td>4</td>
</tr>
<tr>
<td>5</td>
</tr>
<tr>
<td>6</td>
</tr>
<tr>
<td>7</td>
</tr>
<tr>
<td>8</td>
</tr>
<tr>
<td>9</td>
</tr>
<tr>
<td>10</td>
</tr>
</tbody>
</table>

Figure 6.12: The effect of search on the test case population. For each search objective, the values for the initial population are marked with “Initial”, and those from the final population with “Final”. Note: each search objective is shown to a different scale.
Figure 6.13: The effect of interaction on the test case population. For each search objective, the values for the non-interactive execution are marked with “N-int”, and the values for execution including interaction are marked “Int”. Note: The scale for each search objective is different. All values are for the final population at the end of both the interactive and non-interactive runs.
The null hypothesis, that the samples come from the same population, was rejected for all the search objectives with \( p < 10^{-5} \). The values can be seen in Table 6.4. We can confidently conclude that the search component has a considerable effect on the outcome of the ISBST tool.
To assess the impact of the interaction, we compared the populations of test cases resulting from the non-interactive execution against the test data resulting from the experiment. Two comparisons were conducted on an objective by objective basis, and the values for the two populations can be seen in Figure 6.13. Note that interaction seems to be further from the optimum than running with a fixed set of objectives. A non-interactive run supposes that all the objectives, and their relative importance, are known in advance and fixed, which is difficult to achieve a priori. Thus, changing the weights is detrimental to the search if we compare against an unachievable ideal run, where the “correct” weighting is known in advance.

We distinguish between Interaction Significance (Actual) and Interaction Significance (Potential), both seen in Table 6.4. For the Interaction Significance (Actual), we compared the test cases exported by the participants in the experiment against those developed by the ISBST system without any interaction. This has the benefit of comparing the test cases that the participants though best against the non-interactive run, but gives little information about how the two test case populations compare.

To compare the entire population produced by the participants using the ISBST system against that developed by the ISBST system without the benefit of interaction, the Interaction Significance (Potential) was calculated, and is seen in Table 6.4. The Interaction Significance (Potential) allows us to analyze the test cases that were developed, not just those that were exported by participants. This addressed the concerns that participant inexperience might result in relevant test cases not being exported. The Interaction Significance (Potential) provides more information on the overall difference between the two populations of test cases, and lowers the impact of the participants’ choice of exported candidates.

In spite of the overall variance in the interaction strategies and approaches used by the participants, the interaction component clearly has a significant impact on the overall outcome.

A closer inspection of data available for individual participants provides more evidence to support the notion that some of the objectives are more intuitive and easier to optimize. It also emphasizes the importance of search objective definition in the application of the ISBST tool.

### 6.4.3 Fatigue

To answer RQ3, we asked each participant to fill in the NASA Task Load Index after completing each of the two 45-minute sessions. The Mann-Whitney U Test
Table 6.5: The differences between the means of the NASA-TLX values for the manual and ISBST tool. The positive values show that the ISBST tool is more demanding than the manual technique, with the statistically significant differences are marked in bold.

<table>
<thead>
<tr>
<th>Session</th>
<th>Mental Demand</th>
<th>Effort</th>
<th>Temporal Demand</th>
<th>Frustration</th>
<th>Performance</th>
</tr>
</thead>
<tbody>
<tr>
<td>After the first 45-minute session</td>
<td>2.6</td>
<td>3.12</td>
<td>2.4</td>
<td>0.38</td>
<td>0.13</td>
</tr>
<tr>
<td>After the second 45-minute session</td>
<td>1.03</td>
<td>2.33</td>
<td>1.93</td>
<td>2.31</td>
<td>0.9</td>
</tr>
</tbody>
</table>

was performed on the results to determine if any of the differences observed were statistically significant.

The results show that, overall, the participants perceived the ISBST tool to be more demanding on their mental, effort, and temporal resources. It is interesting to note, however, that in terms of the frustration engendered by the each of the methods, and of the performance (as perceived by the participants themselves), the differences were not statistically significant.

After the first session, the ISBST system was assessed as being more mentally demanding ($p = 0.013$) and more demanding in terms of Effort ($p = 0.009$). The data collected at the end of the second 45-minute session shows an interesting effect: only the difference in terms of Effort is still statistically significant ($p = 0.022$).

At the end of the second session, each participant had had the experience of using both methods, and was, thus, in a better position to compare them. A more detailed view of the results can be found in Figure 6.14, and difference between the means is shown in Table 6.5. The statistically significant differences are marked in bold.

As a result, we can conclude that the answer to RQ3 is that the ISBST tool is, indeed, more demanding of the domain specialist, at least initially. It is worth noting, however, that the increased effort did not result in significantly higher frustration, and resulted in similar performance.

An additional analysis was also conducted to determine if the order in which participants applied the methods had an effect on the outcome. The analysis was conducted on a dimension by dimension basis, also the Mann-Whitney U test was used for the comparison, and effect sizes were calculated and inter-
Figure 6.14: The level of demand the participants felt was placed upon them by the two methods and the perceived level of performance, as self-assessed using the NASA-TLX after the second 45-minute session. All measurements are on the same scale. The results for the ISBST tool are marked with A - automated, and those for the manual technique are marked with M - manual.
6.5 Discussion

Our findings suggest that the ISBST system and the manual exploratory testing technique investigate different regions of the SUTs behavior space. Therefore, we conclude that the two methods are complementary and that the diversity of the test suite can be increased by combining relevant test cases produced by the two methods.

We assess the two methods in terms of the regions of the behavior space that they cover, and therefore, in terms of diversity. We cannot state, in general, that this is a good measure for the number of faults found by the test suite. However, there is evidence to suggest that increased diversity does result in more fault-revealing test suites [74, 75].

A closer look at the data shows that the test cases developed by the ISBST tool seem somewhat more concentrated, while those developed by the manual technique seem more diverse. Thus, it would seem that adding the test cases produced by the ISBST system would only result in a small increase in the diversity within the test case population. We will argue that the test cases developed by the ISBST tool are a useful addition to the test suite.

The first issue that needs to be discussed is that of relevance. The requirements we defined to develop and analyze the test cases, as well as each of the direction of each search objective, were arbitrary. This makes it difficult to

Table 6.6: The \( p \)-values of the Mann-Whitney U test, for the differences between the two participant groups. Effect sizes were calculated and interpreted using the Vargha-Delaney A measure. Values that are statistically significant marked in bold. Note that all the values have a negligible effect size.

<table>
<thead>
<tr>
<th>Test</th>
<th>Number of Clusters</th>
<th>Number of Iterations</th>
<th>Mean Silhouette</th>
<th>Silhouette Range</th>
<th>Mean Weight</th>
<th>Weights Range</th>
</tr>
</thead>
<tbody>
<tr>
<td>p-value</td>
<td>(&lt;10^{-5})</td>
<td>(&lt;10^{-5})</td>
<td>0.533</td>
<td>0.011</td>
<td>(&lt;10^{-5})</td>
<td>(&lt;10^{-5})</td>
</tr>
<tr>
<td>Effect Size</td>
<td>0.5486</td>
<td>0.5454</td>
<td>0.4947</td>
<td>0.5215</td>
<td>0.4496</td>
<td>0.4394</td>
</tr>
<tr>
<td>Interpretation</td>
<td>negligible</td>
<td>negligible</td>
<td>negligible</td>
<td>negligible</td>
<td>negligible</td>
<td>negligible</td>
</tr>
</tbody>
</table>

As a result of this analysis, we can conclude that the order in which the participants applied the methods, and therefore the distribution of participants to the groups, has no effect on the outcome of the experiment.
6.5 Discussion

assess how relevant the test cases are. But it is worth noting that many of the manually developed solutions that are far from the automated set, are also test cases that the ISBST system would evaluate as having low fitness.

As mentioned in the previous sections, the search-based system has a set of search objectives, each objective with a direction, that it uses to optimize. Participants could choose which of those objectives to favor, but all selected objectives would be optimized according to those directions. This is a limitation that the manual technique did not have, allowing participants greater freedom to explore test cases that the ISBST system would have dismissed as having too low a fitness value.

The evidence suggests that, when faced with a clear set of objectives, the ISBST system is better at evolving solutions that get close to those objectives. As discussed in Section 6.4, there are dimensions where a clear front of non-dominated solutions is obtained, and that front is closer to the optimum for solutions developed by the ISBST tool. Such a method, however, is more vulnerable to incomplete and inaccurate objectives.

The set of objectives chosen for the optimization, however, may be incomplete. Practical experience and new data informs the choice of optimization objectives that a domain specialist might consider relevant at any one time. As a result, identifying the set of optimizable behaviors is a task specific to each individual domain or problem, and not easy to automate. The ISBST system is designed to allow for the addition and removal of behaviors, as appropriate in particular situations. The current prototype, however, places certain limitations on what form new objectives can take and requires a certain amount of experience to implement the new objectives. This means that it is not optimized for domain specialists, and so the cost of adding new behaviors is quite high. A production version of the ISBST tool would include support for the definition and use of custom and domain specific objectives, but additional work is still required before we can clearly state what form such support should take.

This supports the notion that the two methods are complementary, rather than competing. Focusing on the objectives does, however, limit the ISBST tool. If validating the search objectives is not possible or not practical, alternative mechanisms should be found, that can conduct a search by focusing on exploration rather than optimization. There is a potential to improve the ISBST tool by incorporating mechanisms to increase diversity, e.g. Novelty Search [42], Viability Evolution [43], or MAP-Elites [44].

In analyzing the data regarding RQ2, we note that some behavior attributes show greater variation or greater improvement. This can be due to several causes.
First, the difficulty in optimizing each attribute varies, as does their scale. Some attributes have larger scales, where changes in values are more easily observed. Some objectives are easier to optimize and, therefore, can reach more obvious improvements with less effort. Moreover, objectives are sometimes contradictory. The Number of Cluster objective is a good example, as less optimal values are needed for that quality objective in order to obtain better fitness overall.

In addition, the interactive component highlights objectives that show the most improvement. This can be compounded by the interactive component, as objectives that show the most improvement will also draw more attention from the domain specialists and receive more of their time and focus. The data we presented and discussed in specific to the SUT used in this study, but we expect that in any domain, such differences will arise.

Note also that, as the data analyzing the difference between interactive and non-interactive search, in Section 6.4.2, shows that the interactive search seems to be worse overall than the non-interactive search. This is to be expected, as knowing all the objectives ahead of time, and having that selection and weighting stable throughout the search leads to better optimization. As mentioned before, however, coming up with a stable set of objectives, properly weighted is difficult to do \textit{a priori}. Moreover, as the search progresses, the relative importance of objectives may shift, or new search objectives may be added. Thus, the reason for introducing interaction is giving the ISBST tool the flexibility to adapt to changes in objectives or in their relative importance.

Our discussion of fatigue, and our answer to RQ3, showed that participants did consider the ISBST tool to be more demanding mentally and in terms of effort and time requirements. First, this is to be expected, as the added complexity of the search increases the distance between the participants’ actions and their effect on the SUT’s behavior and the resulting test cases.

In spite of the added fatigue, however, we would like to note that all participants were able to use both techniques to create test cases. Therefore, we surmise that the increased demands placed on the participants by the ISBST tool did not prevent the participants from completing their tasks.

The participants also reported higher Effort demands for the ISBST system if they used that system first. While the difference is not statistically significant, further work into the psychological side of ISBST is required before we can confidently draw conclusions on the factors affecting the interaction.

The main factor guiding the choice of SUT for this experiment was the assumption that interaction, and therefore the rationale for the ISBST system, is needed where a human user can contribute their experience and knowledge.
For the ISBST to be beneficial, a problem would have to allow a human user to quickly assess how appropriate a solution is. This is possible where test cases can be quickly visualized, e.g. the k-means algorithm described here. We surmise that the ISBST tool can be adapted to problems where a clear and informative visualization of test cases can be provided, though it is hard to state what shape that visualization will take. Further work is required to determine the types of problems that are best suited to ISBST.

### 6.6 Threats to Validity

#### 6.6.1 Construct Validity

First, we propose the ISBST system as a complementary technique, to quickly explore the behavior space, based on a set of desired behavior attributes. Thus, our study focused on diversity of test cases and on assessing how the techniques we used explored different regions of the behavior space. This relies on the assumption that a more diverse set of test cases will also result in better fault finding.

We cannot claim that using the ISBST tool as a complement to existing testing techniques will result in greater fault finding or increased quality. Only that such a tool increases the number of SUT behaviors that are being investigated. There is, however, work to suggest that test case diversity does result in greater structural and fault coverage [75]. Further studies are necessary before any conclusions can be drawn regarding the impact of such methods on software quality in general and on fault finding in particular.

#### 6.6.2 External Validity

A second issue is the choice of SUT. For this study, we wanted a SUT with a high-dimensional input and behavior, and where assessing the quality of the solution would involve human participants. The chosen SUT had to strike a balance: it had to be simple enough to understand and use in the limited time available to the participants. However, it also had to be complex enough to benefit from exploratory black-box techniques and not allow optima to be simply calculated or exhaustively searched. We chose a system with a high-dimensional input and behavior, based on our previous experience with our industrial partner. The k-means clustering algorithm has the complexity of some of the embedded modules our industrial partner works with, without being domain specific.
The results of the algorithm can be described as the behavior used to optimize, but participants also had access to the assignment of points to clusters. This allowed them to interpret a test case candidate both from the objective perspective of the computed behaviors and from a subjective evaluation of whether or not the clustering seemed valid to them. Thus, evaluating candidate test cases could only be done by involving human participants.

Nevertheless, we cannot claim our conclusions can be applied to any SUT. Systems that have different characteristics: with a lower number of inputs or where the behaviors are not dependent on the inputs alone, could show different effects. Moreover, systems that do not require human input to assess candidate solutions might not need the interactive component altogether. Before using this technique in new domains or where such characteristics of the SUTs are not known, further validation is required.

6.6.3 Other Validity Threats

Finally, the choice of participants and their level of expertise are also threats to validity. The participants for the experiment were all students at one university in Sweden, and participating in the Verification and Validation course, part of the Software Engineering Master's program. This may result in the participants not being representative for the domain specialists they stand in for. The participants are software engineers in training. While they do not have the experience, their skills and knowledge are relevant to the experiment. In addition, work by Kuzniarz et al. [85] suggests that conclusions can be drawn if students are less familiar with a new method being proposed than to the standard it is compared against. Moreover, Höst et al. suggest that conclusions drawn from experiments with students can hold if carried out with students in their final years [86]. The study they performed involved students in their fourth year, in their last or penultimate year of their Master's education. The students in our experiment are also Master's students, taking advanced courses in software engineering, and are in the last or penultimate year of their Master's program. Thus, both these assumptions hold in our case. Nevertheless, further validation is required, to assess the impact of experience on the ISBST system.

An added concern is that of domain knowledge. The ISBST tool relies on domain specialists to assess candidate solutions and guide the search, based on their knowledge and expertise. The participants in our experiment were chosen based on their willingness to participate and their availability in the numbers required for the experiment, rather than domain knowledge or experience. The initial survey of the participants also confirms that their knowledge of the prob-
lem domain, self-evaluated, is limited. This suggests that experienced domain specialists may exhibit different behavior and may obtain different results.

Thus, before applying the findings in industry, further validation is required. Prototyping any tools with the domain specialists would provide improvements to the tool itself, as well as greater insight into any domain or context specific limitations that could influence the search.

6.7 Conclusions

In this paper we have presented an experiment comparing an implementation of interactive search-based software testing, the ISBST tool, and a manual exploratory black-box technique, in terms of developing test cases for a given SUT. The SUT in this case was the Julia implementation of a $k$-means clustering algorithm.

**RQ1** was concerned with the degree to which test cases developed by the automated method investigate different regions of the behavior space, thereby increasing the diversity of available candidates.

The experiment has shown that the two methods focus on different areas of the behavior space and enabled participants to create test cases that exercise different types of behaviors of the system. This indicates that the ISBST tool is a useful complement to the exploratory black-box technique: it allows participants to develop test cases that investigate different behaviors and characteristics of the system.

**RQ2** focused on identifying if both the interactive and the search components of the ISBST system contribute to the observed effects.

To answer this question we have conducted a laboratory experiment to evaluate the impact of both the interaction and the search component on the outcome of the ISBST system. We conclude that both components significantly influence the search process and the final outcome.

The subject of **RQ3** was the demand placed on the domain specialists by each of the two methods. We conclude that the ISBST system seems to demand more effort from the domain specialist, at least initially, but does not result in a significant increase in frustration or a significant degradation of performance.

An additional conclusion is that there are limitations to the objective-based approach that the ISBST tool uses to guide the search. The objectives used, if incomplete or improperly formulated, can be biased against certain types of behaviors and thus limit the search. This is not a problem if the objectives can be extensively validated with domain experts and constantly updated. If such
validation is not possible or not affordable, an alternative method, e.g. non-competitive or exploration focused evolutionary computation, could provide a way to mitigate the limitations of ISBST. Future work will focus on assessing exploration focused algorithms and investigating the benefits they may provide.

6.8 Appendix

Participant Characterization Survey

The experiment began with a participant characterization survey, containing a number of questions aimed at assessing the participants’ experience and knowledge of several key areas: general programming knowledge, industrial experience, knowledge of software testing, familiarity with SBSE, knowledge of statistics, and experience with the domain chosen for the system under test. These factors were self-assessed by the participants and the results are shown in Figure 6.15.

It is worth pointing out that the scales for each of the assessed dimensions were different. The results of the pre-test can be interpreted as follows:

- **General Programming Experience.** This was assessed in terms of the number of programming courses the participants had taken up to the time of the experiment. The values are 1 - for one or two courses, 2 - for 3 or more courses, with higher number for practical experience in industry. Two thirds of the participants had had one or two courses (value 1 - in Figure 6.15), with the remaining third having had more than 3 programming courses (a value of 2 in Figure 6.15).

- **Industrial Experience in Programming.** For this dimension, answers range from 1 - no industrial experience programming, to 3 - more than one year of industrial experience. Most participants to the experiment were students with no industrial experience.

- **Software Testing Experience.** This dimension assesses the experience participants had with software testing before the Verification and Validation course, with 1 representing very little to no experience in testing (even in courses), and 4 representing more than 1 year of industrial experience with testing activities. Most participants had not undertaken any explicit and systematic testing activities. We can only conclude that the Verification and Validation course was the first contact most participants had with systematic testing activities.
- **Experience of Search-based Software Testing (SBST).** The answers for this dimension range from 1 - never heard of SBST before, to 4 - practical experience using SBST. Most participants had attended one lecture on SBST, that was part of the Verification and Validation course, and their answers reflect this.

- **Statistics.** This dimension evaluates the participants’ experience with statistics and the use of statistical methods. The answers range from no knowledge of statistics to practical experience using statistical methods to solve problems. Except for a few outliers, most participants had courses describing statistical methods, but little practical experience.

- **Domain Familiarity.** This dimension evaluates the participants’ familiarity with the domain, in this case with clustering in general and $k$-means clustering in particular. Most participants had no experience at all with clustering, with one participant having used clustering as a means of data analysis.
Figure 6.15: Results of the participant characterization test. The plot centers on the middle (between response 2 and 3). The percentages on the left are the sum for responses 1 and 2, and percentages on the right are the sum for responses 3 and 4.
Chapter 7

Using Exploration Focused Techniques to Augment Search-Based Software Testing: An Experimental Evaluation

Bogdan Marculescu, Robert Feldt, Richard Torkar

7.1 Introduction

Search-based software testing (SBST) applies meta-heuristic search algorithms to problems in software testing [8, 9]. In practice, this means using search-based techniques to optimize quantifiable aspects of the behavior of the system under test (SUT) and/or of the test cases themselves. These aspects are defined in terms of search objectives and there is often a multitude of them. For example one objective might be full (structural) code coverage, another one to reach a high mutation score and a third one that the test case is as short as possible.

Lehman and Stanley [87], draw attention to the limitations of search-based systems that are based solely on the optimization of objectives, particularly in
the case of deceptive problems. They define a deceptive problem as a problem where one “must seemingly move farther from the goal to ever have the hope of reaching it.”

Feldt and Poulding [88] also call for a broadening of the existing notion of search-based technique and support that argument with their finding that Genetic Algorithms (GA), either in their single-objective or multi-objective forms, are the most prevalent algorithm applied in recent work in software testing.

In the general case, the problem of exploring the behavior space becomes a many-objective problem. Ishibuchi et al.[89] define a many-objective problem as a multi-objective problem with four or more objectives.

Our previous work on developing and industrially evaluating an Interactive Search Based Software Testing (ISBST) tool [60, 73] has also focused on objective-based optimization. In that case we relied on domain expertise provided by our industrial partner to provide meaningful objectives and to validate our ISBST tool. This ensured that the objectives selected for the ISBST system were relevant to the domain, and as complete as possible in describing the interesting characteristics of the tests and the SUT.

However, such detailed knowledge of the domain may be unavailable in other companies or prohibitive in terms of the cost or time involved. In such cases, the need arises for an exploration-focused approach that can investigate the behavior space of a SUT with little prior knowledge of the topology of that space. Moreover, if the relationship between the inputs of the SUT and its behavior or objective is non-trivial, we can thing of the SUT as a deceptive optimization problem. Then the search cannot target higher fitness areas of the search space, since the position, and even existence, of such areas cannot be known a priori.

We define the “codomain” of a SUT as the set of all possible characteristics of that SUT: all the outputs, or functions of those outputs, are included in this set. Further, we define “observed behavior space”, or just “behavior space”, as the subset of the codomain into which the measured outputs of the SUT, or any functions of those outputs, is constrained to fall. It differs from the codomain in that the behavior space only refers to those outputs that are measured or the functions of those outputs that are computed. Additional behavior dimensions may exist that are not measured, not calculated from other measurements, and potentially not even known. As such, the behavior space does not completely describe all the characteristics of the SUT and may be incomplete. Moreover, the mechanisms used for the measurements may be flawed. The result is that the behavior space is fluid, as new dimensions may be added and new values observed.
7.1 Introduction

This creates a problem for objective-based SBST algorithms, since only optimizing those behavior dimensions that are being measured can lead the search away from interesting behaviors. This issue is even more problematic, as what constitutes “interesting behavior” might not be known from the outset, and behaviors can be more or less interesting as the search progresses and more information becomes available. Moreover, using incomplete behavior dimensions is a difficult problem to identify when using an objective-based algorithm. Therefore, any bias induced by the existence of additional relevant behavior dimensions is also difficult to ascertain.

In an ISBST tool, what is needed is a general way to explore the behavior space and the properties of the generated test cases when it is not yet clear what are interesting or even good behaviors that we are looking for.

In recent years a number of such, exploration-focused algorithms have been proposed and in this paper we evaluate their relative merits for interactive, search-based software test generation: Novelty Search [42], Viability Evolution [43], and MAP-Elites or Illumination Search [44]. These algorithms will be compared with an objective-based alternative: a Differential Evolution (DE) algorithm [49]. Differential Evolution was used in previous validations of the ISBST tool in both an industrial and a laboratory setting, so we chose it as a baseline to evaluate exploration-focused alternatives.

In this paper, we show that objective-based and exploration-based algorithms investigate different areas of the test behavior space. Since they cover a wider area of this space, exploration-based algorithms can be used as methods to investigate and maintain population diversity, and therefore enhance existing search-based software testing techniques. Moreover, by enhancing existing techniques with exploration-based algorithms, they would be less vulnerable to incompletely or incorrectly defined search objectives, which is often the case in real-world, industrial testing problems.

In software testing terms, this means that the two approaches identify different behaviors of the SUT and a combined test suite would have greater diversity. Work by Feldt et al. suggests that higher diversity in test suites is linked to higher structural and fault coverage [74, 75].

Next, Section 7.2 presents the research questions. Section 7.3 provides more detailed information on the context, the implementation of the algorithms included in the study, considerations on how the resulting data was analyzed, and details on the practical execution of the experiment. The results are discussed and analyzed in Section 7.4, and their implications are discussed in Section 7.5. Sections 7.6 and 7.7 conclude the paper and describe our ideas for future work.
This paper studies the following research questions:

**RQ1.** To what extent do exploration-focused algorithms investigate a different area of the behavior space than the objective-based ISBST tool?

Our hypothesis is that exploration-focused algorithms will tend to investigate a different area of the behavior space than the objective focused approach. Intuitively, this is a result of a lack of pressure to optimize the objectives. If this hypothesis holds, we can use exploration-based algorithms to maintain diversity in the solution population and make the overall SBST system less vulnerable to incomplete or inaccurate behavior objectives. Essentially, instead of requiring the tester to know up front which testing objectives to fulfill and set specific targets for each of them, an exploration-based system can automatically explore the space of system and test behaviors and present a more varied set to the tester from which he/she can then choose.

An interesting question to investigate is that of relevance of the behavior areas investigated. Since an exploration-focused algorithm will have a different set of pressures on the population of test case candidates, it is to be expected that it will investigate different behaviors. Comparisons of candidate solutions in multi-objective problems can be done by means of Pareto efficiency or dominance [90]. It would be interesting to see if all of the behaviors found by means of exploration are dominated by those found by objective-based search.

If exploring the behavior space only results in large numbers of dominated solutions, then exploration is only relevant as a first step in investigating a completely unknown behavior space. Conversely, if exploration based techniques still find solutions that are non-dominated, this can mean that the contribution such a technique can bring extends beyond the first look at the behavior space when specific target values are set for the objectives, or some of the objectives.

**RQ2.** Do exploration-focused algorithms provide the same benefit to diversity when running with restricted resources, i.e. a reduced number of available optimization steps.

Our previous studies indicate that there is a practical limit on the resources available to an algorithm in an interactive setting such as ISBST. Long waiting times can lead to the human specialist becoming bored or disengaged, with direct repercussions on the quality of their input and of their guidance of the testing system. Our hypothesis is that exploration-focused algorithms can provide a boost to diversity, even with the limited number of optimization steps available.
If this hypothesis holds, exploration-focused algorithms can be used either to complement objective based algorithms, or to replace them where the latter are inapplicable.

7.3 Experimental Setup

This section will provide an overview of the experiment, along with the research instruments, system under test, methods of analysis, and other practical considerations on the execution of the experiment.

The goal of the experiment is to investigate the areas of the behavior space that each of the algorithms covers and, thus, provide answers to our two research questions. In particular, we want to investigate how the algorithms behave in situations where little initial information is available on the behavior space.

7.3.1 Context

The impetus for this work came out of a previous study [73], describing the Interactive Search-Based Software Testing (ISBST) tool. The ISBST tool uses interaction to enable a human domain specialist to guide the search process by weighting a number of objectives based on their relevance at any given time. That work, however, relied on the experience and knowledge of domain specialists to define and validate the search objectives. In practice, this means that the technique is difficult to apply in situations where such detailed domain knowledge is unavailable.

For a more general application of ISBST, we looked at techniques that allow automated exploration of the behavior space, with little a priori knowledge of its topology or any domain specific limitations.

The information that is initially available on the behavior space is the number of behavior dimensions. Additional information will be obtained as the search proceeds and new values of each of the behavior dimensions will be observed. Thus, all the information that is obtained, is derived from the exploration itself, and can be considered reliable.

Thus, even if the search objectives are incomplete, the focus on exploration diversity means no potentially interesting SUT behaviors are ignored by the search. In the case of inaccurately defined search objectives, the added population diversity means that relevant behaviors are maintained in the population until the problem is identified and corrected.
<table>
<thead>
<tr>
<th>No.</th>
<th>Objective</th>
<th>Description</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>Number of Clusters</td>
<td>Number of clusters to be found. (The ( k )-means algorithm required the number of clusters to be found as an input).</td>
</tr>
<tr>
<td>2</td>
<td>Total Cost</td>
<td>A measure of the distance between the center of a cluster and each element in a cluster. Low values indicate well-defined, tightly packed clusters.</td>
</tr>
<tr>
<td></td>
<td></td>
<td>High values indicate loosely-defined and indistinct clusters.</td>
</tr>
<tr>
<td>3</td>
<td>Number of Iterations</td>
<td>( k )-means clustering is done in several iterations, until the clusters are stable.</td>
</tr>
<tr>
<td>4</td>
<td>Mean Silhouette</td>
<td>The Silhouette is a quantitative way to measure how well each item lies in its own cluster. The Silhouette of each point has a value between 0 and 1, with higher values (closer to 1) indicating that the point lies well within its own cluster and there is no meaningful alternative cluster it could be assigned to. The mean is computed to provide an overview of how well the points belong to their respective clusters across the entire population.</td>
</tr>
<tr>
<td>5</td>
<td>Silhouette Range</td>
<td>This is the absolute distance between the lowest and the highest silhouette values found in the current candidate. A high value for this attribute means that the test case contains both well-defined and ill-defined clusters. A low value indicates that the test case contains only one of the two options.</td>
</tr>
<tr>
<td>6</td>
<td>Mean Weight</td>
<td>The weight of a cluster is the sum of the weights of all the points within it. For the purpose of this experiment, each point has the same weight: ( \text{weight} = 1 ). Larger clusters, with more widely dispersed points, will get high values for this quality objective. Small, tightly packed, clusters will get low values for this objective.</td>
</tr>
<tr>
<td>7</td>
<td>Weight Range</td>
<td>Test cases containing a combination of large and small clusters will obtain a high value with respect to this objective.</td>
</tr>
</tbody>
</table>

Table 7.1: Overview of the quality objectives used in the experiment.

7.3.2 System under Test

Previous experience also informed our choice of system under test. The selected SUT would have to use a large number of inputs and its behavior would have to be expressed as a many-objective problem.

The system under test selected for the experiment is a \( k \)-means clustering algorithm implemented in the Julia programming language \(^1\). The SUT itself is part of the Julia Clustering package \(^2\), and is actively maintained by its developers.

For the purpose of the experiment, each of the investigated search algorithms will search for a group \( n_{\text{inputs}} = 61 \) inputs, representing a group of \( n_{\text{points}} = 30 \) two-dimensional points, and the \( n_{\text{clusters}} \) number of desired clusters. The algorithm will employ the SUT, the \( k \)-means algorithm from the Clustering package, to create \( n_{\text{clusters}} \) clusters from the input points.

\(^1\)http://julialang.org
\(^2\)https://github.com/JuliaStats/Clustering.jl
7.3 Experimental Setup

The behavior space consists of \( n_{\text{objectives}} = 7 \) dimensions that characterize the resulting clustering. The behavior dimensions are described in more detail in Table 7.1. The behavior dimensions are based on the clustering validation information already provided by the implementation of the \( k \)-means algorithm; this does represent a use case where little is really known about the behavior space. The developer/tester might not know which test cases and output behavior that are valid and important for the testing to be of high quality. One potential solution consists of one set of inputs and a set of values for each of the behavior dimensions.

For the objective-based system, each search objective can be written to be maximized or minimized. In our case, we wrote the search objective so that smaller values are better, for all cases. This is done solely to simplify presentation and interpretation of the results.

Note that the number of clusters is both an input and a search objective. An input, because the number of clusters is a requirement of the \( k \)-means algorithm. However, it is difficult to know \textit{a priori} how many clusters are needed. So the number of clusters is to be minimized, but test cases with a higher number of clusters may provide benefits with respect to other objectives. The number of clusters must be part of the fitness evaluation and have an influence on the fitness score, for the system to be able to make such trade-offs.

The Clustering implementation we use as SUT is a good representation of the system that our industrial partner commonly uses: it has a large input space, a many-objective behavior space, and is too vast to be explored manually.

For this evaluation, we have also purposefully ignored domain knowledge about the values and limitations of the behavior dimensions. This would recreate the general case scenario discussed earlier, where the only information regarding the behavior space that is available is the number of dimensions. The practical effect of this decision is to allow exploration-focused algorithms the freedom to investigate the behavior space at will, and provide no negative impact on the objective-based system.

7.3.3 Research Instruments

All the search algorithms being compared were implemented in the Julia programming language, to ensure that no bias be traced to language specificities. Moreover, this also provided confidence that interfacing with the SUT would not provide additional complications, either during development or while running the experiment.
All the algorithms use the same mechanism to generate an initial population of test cases, and the initial population is the same size for all algorithms: \( N_{\text{population}} = 100 \). The initial population is generated randomly from a uniform distribution, and covers the entire, allowed input space. For this experiment we consider an initial population with floating point values \( V_{\text{coord}} \in (-1000.0, 1000.0) \) for the point coordinates, and with \( n_{\text{clusters}} \in (3, 11) \).

The exploration-focused algorithms are all evolutionary algorithms and need genetic operators to be applied. To only compare their ability to explore the spec of test behaviors we used the same mutation operators for all of them. The candidate selected for mutation has one of the input dimensions mutated. The mutation is based on a normal distribution, with a high likelihood of a small change, but with larger changes possible. The mutation is bounded by the extrema of the input dimension, so no invalid inputs are obtained.

Note that, while the mutation mechanism is the same for all the algorithms, selecting which candidate to mutate varies and will be explained in more detail for each algorithm.

The stop condition for all the algorithms will be reaching a maximum number of optimization steps \( n_{\text{steps}} \). The value of \( n_{\text{steps}} \) is the same for all the algorithms, thus ensuring a fair evaluation of their capabilities.

The four algorithms that are being evaluated are presented below.

**Viability Evolution.**

Viability Evolution [43] is a technique that allows users to specify a set of desired values of the behavior dimensions, that the algorithm will evolve towards.

For each dimension of the behavior, a set of boundaries is defined. The initial boundaries are selected to encompass the entire population. The boundaries are then updated to exclude a fraction \( f_{\text{excluded}} \) of the candidates. In our implementation, each boundary update renders at least \( f_{\text{excluded}} = 0.33 \) of the population unviable.

For each optimization step, one of the remaining candidates is selected and mutated. If the mutant is viable, it is added to the population. If the mutant is not viable, it is discarded. Once the population numbers have returned to previous levels, the boundaries are again updated and the process resumes. The algorithms stops when the current population boundaries match the target behavior boundaries, or the maximum number of optimization steps has been reached.

Viability Evolution employs a family system as a mechanism to ensure population diversity. Each candidate in the initial population becomes the progenitor of a family. Each mutant resulting from the candidate is part of the same fam-
7.3 Experimental Setup

ily. Selecting which candidate to mutate is based on the family. The chance for a family to be selected is inversely proportional to its size. Once a family has been selected, a candidate for mutation is picked randomly, from a uniform distribution.

As stated previously, the behavior of the SUT is defined in terms of dimensions that are to be minimized. This enables us to simplify the definition of the target behavior for Viability Evolution. It also means that the algorithm will not stop until the maximum allowed number of optimization steps $n_{\text{steps}}$ has been reached.

It should be noted that some familiarity with the behavior space is required, to allow a target for Viability Search to be defined in a meaningful way.

**Novelty Search.**

Novelty Search, is defined by Lehman and Stanley as an evolutionary algorithm that differs from the norm by “replacing the fitness function with a novelty metric” [87]. Solutions that are novel, are then added to an archive for future comparisons. However, defining a meaningful novelty metric is not as simple a task as it sounds. The initial paper describes a maze navigating robot, and defines the novelty metric as the Euclidean distance between the point the current solution reaches, and the points reached by previous attempts, as stored in the archive. Since the authors are looking at behavior in a two-dimensional space, Euclidean distance is a reasonable choice.

In our case, we assume little knowledge of which of the dimensions of the behavior space is relevant. As a result, we would like to evaluate SUT behavior based on all identified dimensions. This turns the relevant behavior space from two-dimensional to many-dimensional. For many-objective problems, however, work by Aggarwal et al. [91] suggests that Euclidean distance is less meaningful. Instead, they suggest fractional distance metrics

$$ \text{dist}_{d}^{p}(x, y) = \left[ \sum_{i=1}^{d} ||x^i - y^i||f \right]^{1/f} $$

In the formula above, $d$ is the number of dimensions and $f$ is of the form $1/l$, where $l$ is some integer. Their work argues that fractional distance metrics is more appropriate for high-dimensional data. In our implementation of Novelty Search we used the fractional distance metric described above. The value, $l = 7$, was based on the number of behavior dimensions.

Novelty Search computes the sparseness for each candidate, as the distance to the $k$-nearest candidates. The sparseness is then used for transferring candidates to the archive and as a mechanism to select which candidates are to be mutated.
The candidates with the highest sparseness are those considered for inclusion in the archive. A candidate thus selected, is then assessed with respect to its distance from the other candidates in the archive. If the solution between the current candidate and the closest candidate already in the archive is higher than a threshold value $\rho_{\text{threshold}}$, the current candidate is included in the archive. The first member of the archive is the candidate with the highest sparseness in the initial population. The value of $\rho_{\text{threshold}}$ is based on the distance between the initial member of the archive and the most distant candidate from it, i.e. the highest observed distance in the initial population.

The likelihood that a candidate is selected for mutation is directly proportional to the sparseness value of that candidate.

As with the other algorithms, the Novelty Search algorithms will use the maximum allowed number of optimization steps as its only stopping condition, to ensure a fair evaluation.

We also note that Novelty Search requires that a meaningful measure of “novelty” be provided. This suggests that some understanding of the behavior space is required to ensure correct application of the algorithm. However, a general novelty metric such as the one we use here might be suitable in many cases given a vector of values for the behavior dimensions.

**MAP-Elites (Illumination Search).**

MAP-Elites is an algorithm proposed by Mouret and Clune [44] to explore a search space while seeking to avoid local optima. Since they define their algorithms as “illuminating search spaces”, we have taken to calling it Illumination Search.

The algorithm splits the behavior space into a number of cells, each cell holding at most a single candidate. At each optimization step, a cell is chosen randomly, from a uniform distribution. The candidate within the selected cell is mutated, it is supplied as a test case to the SUT and the behavior recorded. The appropriate cell for the mutant is then found. If that cell is empty, the mutant is assigned to occupy it and the process resumes. If the cell is occupied, the mutant replaces the existing occupant only if its performance is better than that of the occupant. The algorithms then resumes. Note that it needs a definition of performance in addition to the definition of behavioral dimensions.

A few things need to be discussed regarding Illumination Search, and our implementation of the algorithm. First of all, defining performance is not a trivial task, given the multi-dimensional behavior space. We settled on defining performance as Pareto dominance. Thus, a mutant replaces the current occupant of a cell if it is Pareto dominant. In this case, this was possible since information on what would constitute a “better” candidate is available for each dimension.
7.3 Experimental Setup

In situations where this information is not available, alternative measurements of performance would have to be defined.

The second issue is that Illumination Search divides the behavior space into cells, but the exact mechanism is not described in detail in the original paper [44]. In our work, we assume limited knowledge of the behavior space, meaning that theoretical extrema cannot be defined for each dimension. Our solution to this problem was to define cell size in terms of the random population generated at the beginning of the algorithm. This is a solution which is similar how Novelty Search decides the value of its threshold from the initial population.

Two versions resulted from this approach. The first divided the space defined by the initial population, and split that into several cells, adding a cell for higher and a cell for lower values for each dimension. This first version provides good resolution for the behavior area covered by the initial population, and allows the search to cover the initial area and identify good solutions within existing maxima and minima. The drawback is that searching outside the extrema of the initial population is problematic with this approach, effectively limiting the search. The search could thus be stymied by an initial population that does not well represent the whole set of behaviors that can be found.

The second version used the initial population to define a cell size, and uses that cell size to classify further output, even outside the min and max values for each dimension as seen in the initial population. This places no limit on the number of cells that can be investigated and does not limit the search. The goal is to preserve the underlying philosophy of MAP-Elites, which is to ensure that each potential solution competes against similar candidates.

Note that, given the way we implemented Illumination Search, the population used to calibrate the cells is an essential factor. Using a more diverse set of candidates to define the cells will likely result in a faster exploration of the behavior space. This resulted in two extra subversions, one using only the initial population of $N_{population} = 100$ candidates to calibrate the cells. The other uses the cumulative population of all the algorithms (investigated prior to Illumination Search) to allow for the purpose. In practical terms, the first is equivalent to the algorithm running on its own. The second is representative of running the algorithm as part of a large system, with other techniques available to provide the missing information.

**Differential Evolution.**

The traditional, objective-based search algorithm that forms the basis for our comparison is a Julia implementation of the ISBST tool and its Differential Evolution search algorithm [73]. The tool presented there consisted
of an interaction focused component, the *Outer Cycle*; and a search-based component, the *Inner Cycle*. Since the *Outer Cycle* is concerned with interacting with the domain specialist, it has no impact on the work presented here.

The *Inner Cycle* consists of a Differential Evolution algorithm [49]. Differential Evolution is a parallel, direct search method. Each potential solution is a vector of real numbers. The initial population is chosen randomly from a uniform distribution, and covers the entire parameter space. New parameter vectors are added by mutation: adding the weighted difference between two population vectors to a third vector. For each target vector $x_{i,G}$, where $i = 1, 2, \ldots, N_{\text{population}}$, a mutant vector is generated as follows:

$$v_{i,G+1} = x_{r_1,G} + F \cdot (x_{r_2,G} - x_{r_3,G})$$ \hspace{1cm} (7.1)

where $r_1, r_2, r_3 \in 1, 2, \ldots, N_{\text{population}}$, are integers, and mutually different, and different from the running index $i$. $F$ is a real and constant factor $\in (0, 2]$ which controls the amplification of the differential variation $(x_{r_2,G} - x_{r_3,G})$.

The result $v_{i,G+1}$ is then subjected to crossover, by mixing its parameters with those of another predetermined vector, and the outcome of this operation is called trial vector. If the trial vector is an improvement over the target vector, it replaces it in the following generation [49].

The crossover rate we used is $cr = 0.5$, the scale factor is $F = 0.7$, and the population size is $\text{population} = 100$. The mutation strategy is that proposed by Storn and Price [49]: DE/rand/1/bin. The strategy uses a differential evolution algorithm (DE); the vector to be mutated is randomly chosen (rand); one difference vector is used (1); the crossover scheme is binomial (bin).

To allow the single objective DE to handle multi-objective and many-objective problems, we used the Sum of Weighted Global Averages [50].

This approach normalizes all the values in a generation to an interval between the largest and the smallest values observed for a given objective, both in the current and previous generations. Each solution is assessed and receives a score for each of the quality objectives. The weights are then used to combine the scores into a single fitness value for each candidate.

$$IFF(j) = \sum_{i=1}^{n\text{Objectives}} \text{Weight}_i \cdot \text{Value}_{i,j}$$ \hspace{1cm} (7.2)

where $IFF(j)$ is the fitness value of candidate $j$, $\text{Weight}_i$ is the current weight of the objective $i$, and $\text{Value}_{i,j}$ is the fitness value of candidate $j$ measured by objective $i$. The value of $IFF(j)$ is the sum of the weighted fitness values for all $n\text{Objective}$ objectives.
7.3 Experimental Setup

For this study, we leave aside the interactive component and assume no intervention from any human domain specialist. As a result, all the objectives have the same, default, weight:

\[ \text{Weight}_i = 0.5 \]

The ISBST serves as a reference, and data obtained from our previous and ongoing empirical evaluations of the ISBST system is used to calibrate the current experiment. We can thus compare the automated search algorithms in this study also to results from manual interaction with the tool by human testers.

7.3.4 Analysis of exploration results

The goal of our analysis is to determine the degree to which the exploration-focused algorithms investigate the same (or, conversely, different) areas of the behavior space as the differential evolution component of the ISBST. However, the behavior space is vast and complex, so some simplifications were made to enable a more expressive analysis.

The behavior space was divided into cells based on the maximum and minimum values observed in all the populations resulting from all runs of the algorithms. The candidates in the populations were assigned to the cells. Each of the exploration-focused algorithms was compared against the differential evolution algorithm in terms of the number of cells that overlapped, i.e. cells where both algorithms had at least one candidate present, and the number of cells that were exclusively occupied by candidates from one algorithm.

We assess whether an algorithm has explored a different area of the search space from the objective-based approach based on the number of cells that had occupants from that algorithm but not from the DE.

We had concerns regarding the lack of pressure to provide interesting candidates in the exploration-focused algorithms, as opposed to the clear drive to optimize in the objective-based approach. The vast behavior space means that there is no easy way to determine if the extra behaviors that are explored are meaningful. To address this concern we introduced a new measure: the number of candidates found by an exploration-based algorithm that are not dominated by candidates found by DE.
The number of optimization steps between two interaction events in ISBST. The ISBST system performs 250 optimization steps after receiving input from the human specialist. As a result, we use this value to signify the smallest optimization step budget that an algorithm would have available.

Practical experience during the use of the ISBST system indicates that the average number of times that a human domain specialist interacts with the ISBST system in one session is $n_{interactions} = 20$. This value results in a budget of $n_{steps} = 5000$ optimization steps.

The highest number of observed interactions between a human domain specialist and the ISBST system, in one session, was $n_{interactions} = 86$. This results in a budget of $n_{steps} = 21500$.

The equivalent of around 50 specialists interacting with the system for one 45-minute session. This is used as an extreme value.

<table>
<thead>
<tr>
<th>Value</th>
<th>Motivation</th>
</tr>
</thead>
<tbody>
<tr>
<td>250</td>
<td>The number of optimization steps between two interaction events in ISBST. The ISBST system performs 250 optimization steps after receiving input from the human specialist. As a result, we use this value to signify the smallest optimization step budget that an algorithm would have available.</td>
</tr>
<tr>
<td>5000</td>
<td>Practical experience during the use of the ISBST system indicates that the average number of times that a human domain specialist interacts with the ISBST system in one session is $n_{interactions} = 20$. This value results in a budget of $n_{steps} = 5000$ optimization steps.</td>
</tr>
<tr>
<td>21500</td>
<td>The highest number of observed interactions between a human domain specialist and the ISBST system, in one session, was $n_{interactions} = 86$. This results in a budget of $n_{steps} = 21500$.</td>
</tr>
<tr>
<td>250000</td>
<td>The equivalent of around 50 specialists interacting with the system for one 45-minute session. This is used as an extreme value.</td>
</tr>
</tbody>
</table>

Table 7.2: Brief explanation of the optimization steps budget available to each algorithm.

### 7.3.5 Experiment Execution

To ensure that the algorithm comparison is fair, each had the same amount of ‘effort’ available to work with. We define effort as the number of optimization steps available to each of the algorithms. This definition is based on the definition of Črepinšek et al. [72], that suggest measuring evolutionary algorithm performance based on the number of fitness evaluations. In our case, each optimization step results in a single fitness evaluation, for all the algorithms, so the measurements are equivalent. We chose optimization steps because they provide a convenient stop condition for the algorithm runs and because we can more easily relate optimization steps to previous uses of the ISBST and, therefore, to practical experience.

We used four values for the available budget of optimization steps, based on relevant values observed in previous, practical assessments of the ISBST system: $n_{Steps_1} = 250$, $n_{Steps_2} = 5000$, $n_{Steps_3} = 21500$, $n_{Steps_4} = 250000$. A more detailed explanation for the reason for choosing each of the variables can be found in Table 7.2.

The three values for the budget are based on observed behavior of human domain specialists interacting with the ISBST system. While it is difficult to comment on other values, we will state that we have observed users interact with the system in 45-minute sessions without fatigue affecting their behavior. We will use the observed number of interactions over a 45-minute session as the basis for the available budgets for the evaluation.

For each of the budget values, each algorithm was run 30 times.
7.4 Results and Analysis

Throughout this section, we will refer to the different algorithm and algorithm versions by a number of designations. Those designations are clarified in Table 7.3.

As discussed, we base the analysis on splitting the observed behavior space into a number of cells. The observed behavior space consists of the extreme values in the cumulative candidate population for all of the algorithms. We evaluate each algorithm by comparison against the reference: the objective-based DE algorithm.

7.4.1 Exploratory Power

First, a quick overview allows us to see that all the algorithms explore significant areas of the behavior space that do not overlap with those investigated by the objective-based technique.

The comparison between the number of cells that are investigated by the exploration-focused algorithms exclusively, measured in terms of percentage of the total number of cells covered, can be found in Figure 7.1. From these results we can conclude that regardless of the optimization step budget, exploration-focused algorithms tend to investigate a much larger area of the behavior space than objective-based algorithms.

Table 7.3: Designations of the exploration-based algorithms, and (where applicable) their variants.

<table>
<thead>
<tr>
<th>Designation</th>
<th>Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>III,1e</td>
<td>Illumination Search. 1 - with the cells focused on the initial population; e - early, i.e. the cells are based on the initial, random, population.</td>
</tr>
<tr>
<td>III,2e</td>
<td>Illumination Search. 2 - the initial population defines the size of the cell, but does not limit the number of cells; e - early, i.e. the cells are based on the initial, random, population.</td>
</tr>
<tr>
<td>III,1l</td>
<td>Illumination Search. 1 - with the cells focused on the initial population; l - late, i.e. the cell are defined based on the extrema of all the populations seen across all algorithms for the current run.</td>
</tr>
<tr>
<td>III,2l</td>
<td>Illumination Search. 2 - the initial population defines the size of the cell, but does not limit the number of cells; l - late, i.e. the cell are defined based on the extrema of all the populations seen across all algorithms for the current run.</td>
</tr>
<tr>
<td>Novelty</td>
<td>Novelty Search</td>
</tr>
<tr>
<td>Viability</td>
<td>Viability Evolution</td>
</tr>
</tbody>
</table>
Figure 7.1: Areas of the behavior space covered exclusively by the exploration based algorithms, measured in percentage of the number of cells in the behavior space.
7.4 Results and Analysis

Figure 7.2: Number of exclusive and non-dominated candidates developed by each algorithm, expressed at percentages of the total number of occupied cells.
The next step in the analysis is to see how many of these cells are relevant. Since both the search space and the behavior space, are many-dimensional, it is reasonable to expect that there are many areas that an exploration-focused algorithm can explore. It may be worth considering how many of these behaviors are relevant. While it is difficult to assess, we suggest that a potentially interesting candidate is one that is not Pareto dominated by existing candidates. This is computed as follows: for each exploration focused algorithm, a mixed population is created consisting of the candidates developed by that algorithm and the objective-based algorithm. We report the number of non-dominated candidates, from among this mixed population, developed by each exploration focused algorithm. While it is difficult to assess, we suggest that a potentially interesting candidate is one that is not Pareto dominated by existing candidates. This is computed as follows: for each exploration focused algorithm, a mixed population is created consisting of the candidates developed by that algorithm and the objective-based algorithm. We report the number of non-dominated candidates, from among this mixed population, developed by each exploration focused algorithm. The numbers of non-dominated candidates produced by each algorithm can be seen in Figure 7.2, measured as percentages of the total number of occupied cells.

It is worth noting that, while all algorithms seem to investigate different areas of the behavior space from the objective-based system, a relatively small number of the resulting candidates are non-dominated.

This suggests that exploration is a powerful force in terms of investigating behaviors that might not otherwise be observed. Large areas of the behavior space can be investigated, that would not be reached by objective-based means, even in situations where little information is available about the behavior space.

The highest number of non-dominated candidates are obtained by the generous implementations of Illumination Search, that benefit from a more diverse population for calibration. Additionally, Viability Search also produces relatively high number of non-dominated solutions, but this algorithm contains a clear element of optimization that guides the exploration more towards a non-dominated solution front. The downside of these algorithms is that they require a very diverse population, for the former, and a clear set of goals, for the latter.

Of the exploration-based algorithms, the sub-variant of Illumination Search that performs worst in terms of exploring behavior, performs best in terms of generating non-dominated solutions. This seems to indicate that the drive to explore the behavior space and the drive to optimize the objectives are contradicting.
From this, we conclude that exploration-focused algorithms are extremely useful tools for maintaining and increasing population diversity, but less useful when it comes to driving the search towards optima. Given a vast behavior space to explore, it is not unreasonable that these algorithms expend their optimization step budget on expanding the covered area, and thus are less focused on finding optimal values.

The candidate populations developed by most of exploration-based algorithms consist of 80% or more candidates that cover cells of the behavior space that are not covered by the objective-based algorithm.

The one exception seems to be one of the sub-variants of Illumination Search. The same variant has, on the other hand, a higher number of non-dominated solutions than any of its peers. This seems to suggest that the exploration focus is not compatible with the drive to optimize.

Moreover, the lack of focus on optimizing a set of given objectives means that the exploration based algorithms are less sensitive to mistakes in the definitions of those objectives. In situations such as those mentioned above, where validating objective completeness and correctness is not possible, exploration-focused algorithms are useful alternative to objective-based algorithms.

Thus, the answer to RQ1 is that the exploration-based algorithms spend most of their effort investigating areas of the search space that objective-based functions ignore. Each exploration-based algorithms investigates \( n_{\text{cells}} = 650 \) or more cells that objective-based algorithms do not, amounting to 70% or more of the candidates developed.

### 7.4.2 Optimization Step Budget

The second of the research questions posed some concerns on the effect of restricting the available optimization step budget. The hypothesis is that the exploration-focused algorithms exhibit the same benefits even when the optimization step budget is low. This is important to consider given that in our previous evaluations, we determined that the ISBST system’s current Julia implementation could perform around \( n_{\text{steps}} = 250 \) optimization steps between two interactions with the human domain specialist. This number is high enough that the domain specialists will see improvement in the candidate population, but low enough to ensure that they will not become bored or disengaged with using the testing tool: usually amounting to fewer than 10 seconds. This number, however, is implementation specific, as different programming languages may be slower, and thus be able to conduct fewer steps in the same amount of time.
Figure 7.3: The Viability Evolution algorithm behavior with different number of optimization steps available.
Figure 7.4: The Novelty Search algorithm behavior with different number of optimization steps available.

(a) Exclusive cells.

(b) Non Dominated.
Figures 7.3 and 7.4 show a closer look at the behavior of two of the algorithms, compared across several values of the available optimization step budget. The two figures show clear differences between the algorithms.

Viability Evolution, shown in Figure 7.3, shows expected behavior. As the number of optimization steps available increases, more areas are investigated. The growth in the number of cells that are exclusively covered is not proportional to the additional resources available. The number of non-dominated solutions decreases, as the additional optimization steps go towards investigating non-optimal areas of the space. Since the final comparison is done against the entire area covered, and the larger optimization budget is available to the competition as well, a slight drop in the number of non-dominated solutions is to be expected.

Novelty Search, on the other hand, has somewhat unintuitive behavior, seen in Figure 7.4. The area that it covers also grows, and the same diminishing of the growth rate can be observed. However, in the case of Novelty Search, the number of non-dominated solutions drops significantly. We suggest that this is due to the absence of any pressure to optimize in Novelty Search.

We can see, however, that both algorithms perform their exploration duties well, even with the lower budget available. The answer to RQ2, therefore, is that exploration-based algorithms provide benefits to diversity even when running with a small number of optimization steps. This means that such algorithms can be used in interactive search-based systems, in spite of the reduced resources available.

7.5 Discussion

We have shown that exploration-based algorithms can help alleviate some of the problems inherent in attempting to search a behavior space where little information is available.

We started from the assumption that it will not always be possible to ensure the completeness and correctness of search objectives for the ISBST system. This is problematic, as incomplete search objectives mean the search can be steered away from potentially interesting candidates, in our case, potentially useful test cases and behaviors.

As an alternative, we focused on exploration-focused algorithms, to conduct an initial exploration of the behavior space.

It must be noted, however, that all of the exploration-focused algorithms discussed in this paper require some knowledge of the domain and of the behavior space; in practice they cannot work effectively without some information
about the domain. Novelty Search requires a domain-relevant distance to be used. We used a fractional distance metric, as the most general type of distance for high-dimensional spaces. Undoubtedly, the algorithm will perform better with domain-specific distance metrics.

Viability Evolution reduces the space of viable solutions, trying to guide the population towards optimal values for each of the behavior dimensions. As a result, one must know for what would be optimal values for every behavior dimension. In the most general case, one should be aware of which dimensions are to be minimized and which are to be maximized, and use absolute extrema to update the boundaries. We have chosen to implement this most general case, which likely affected the performance of this algorithm.

MAP-Elites, or Illumination Search, requires that the behavior space be split into cells, based on maximum and minimum values. We supposed that such values are not available, and defined four versions of this algorithm. All of the versions we defined are somewhat hampered by the reduced information. The ideal case, where the extrema of each behavior dimension are known, will likely yield better results.

We agree that in trying to make all these algorithms completely domain agnostic, their performance may have suffered. Nevertheless, the algorithms have provided a useful mechanism for exploring a high-dimensional behavior space, even with the information about the behavior space being limited.

The approach we used for analysis should also be discussed. Again, we assumed a situation with minimal knowledge of the behavior space. So the analysis relies on the cumulated final population of all the algorithms. This makes comparison more difficult, as the final population is unlikely to be identical in any two runs.

The final population, however, is the result of accumulating a large number of candidates from all the different algorithms. This ensures a diverse population that covers a large area of the behavior space for the SUT. Moreover, each run was conducted 30 times. Given the large number of diverse candidates in these final populations, and the large number of runs conducted without incident, we would argue that the final cumulated population is stable enough to serve as a reference and to allow for a useful analysis.

These findings open the possibility of hybrid SBST and ISBST systems. Exploration would be used to investigate the behavior space, and to define suitable objectives. Optimal solutions, as defined by those objectives, could then be found and proposed. Exploration-based search could also be used as a mechanism for maintaining population diversity, reducing the risk of objective-based search being stuck in local optima.
Conclusions

In this study, we have compared exploration-focused algorithms, Novelty Search, Viability Evolution, and four versions of MAP-Elites or Illumination Search, against an objective-based algorithm, i.e. Differential Evolution.

We have observed that exploration-focused algorithms can investigate the behavior space, even in situations where there is little information available about that space. Not surprisingly, these algorithms are not as effective in driving towards an optimal solution if the tester is interested in a specific part of the behavior of the test or SUT. However, in situations where the objectives to be optimized are incorrect or incomplete or when little is known about what type of behavior the system can or should have, exploration-focused can still be applied and provide important behavior about the SUT and its tests.

In addition, we conclude that the exploration-focused algorithms provide useful results even in situations where there are few optimization steps available. This ability to explore the behavior space, even with limited resources, offers a useful mechanism for an initial exploration of an unknown behavior space. Typically, of the candidate solutions developed by exploration-based algorithms, more than 80% were not found by their objective-based counterpart. Thus we propose that exploration-focused search algorithms can be an important future component in interactive as well as non-interactive search-based software testing systems.

Future Work

This study was driven by the need to find a way to explore the behavior of complex problems, with little information regarding the topology of that behavior space. For a complex problem, one with high dimensional input and output space, it may also be difficult to ensure the validity, correctness, and completeness of any defined objectives. As a result, exploration-focused algorithms provide a useful means of exploring the behavior space, without being affected by any fault in the defined optimization objectives, or even in the absence of optimization objectives.

We propose, therefore, a hybrid type of search. One where exploration is conducted in parallel with objective-based optimization. In the context of the ISBST tool, the human domain specialist can decide, based on their knowledge and their confidence, whether to explore the behavior space for a particular
SUT, or to define quality objectives and to search for more clearly optimized candidates.

Alternatively, exploration can be a background process, to ensure that the diversity of the candidate population is maintained and that the objective-based optimization can avoid getting stuck in local optima.
Chapter 8

Transferring Interactive Search-Based Software Testing to Industry

Bogdan Marculescu, Robert Feldt, Richard Torkar, Simon Poulding

8.1 Introduction

Search-based software testing (SBST) is the application of optimization algorithms to problems in software testing [8, 9], with new algorithms and approaches being proposed and evaluated. Efforts have been made to ensure that these new approaches receive rigorous evaluations, and benchmarks have been developed to enable comparisons between different approaches and their respective evaluations. One example of developing and evaluating new approaches is our work with the Interactive Search-Based Software Testing (ISBST) system. The ISBST system was proposed [60], was evaluated both in academia [92] and in industry [73], and further refinements have been proposed [93]. Thus, the ISBST system has been evaluated and validated in academia, and preparations for its transfer to industry are ongoing.

Successful transfer of SBST to industry would be enable companies to improve the quality of their software quality assurance process, with limited re-
sources. In addition to being an effective solution to real engineering problems, successful transfer would also have academic benefits, both in terms of the increase in quality and efficiency that existing evaluations claim for SBST, and in terms of generating additional information, validating existing approaches, and refining our understanding of the underlying phenomena.

In this paper, we will use the model of technology transfer to industry proposed by Gorschek et al. [48], henceforth referred to as the Technology Transfer Model or TTM, to evaluate our attempts at transferring SBST to industry, as well as discussing the lessons learned during the transfer process.

This paper will present our work evaluating and validating the ISBST system. We will use the Technology Transfer Model to assess the maturity of the ISBST system and to frame the lessons learned from its development and evaluation. Section 8.2 of the paper discusses related work, Section 8.3 discusses the context, our industrial partner, and describes the artifacts used in the study. It also presents a synthesis of the development and evaluation of the ISBST system within the framework of the Technology Transfer Model. Section 8.4 describes the static validation of the latest version of the ISBST system, on-site, using industrial code and performed by industrial practitioners. Section 8.5 discusses the lessons learned throughout the development and evaluation of the ISBST system, from its conception and up to, and including, the current study. Section 8.6 considers the threats to the validity of the work to develop, assess, and deploy the ISBST system, from its conception until the present version. Section 8.7 discusses some of the implications of the study, and Section 8.8 presents our conclusions.

8.2 Related Work

Search-based software engineering (SBSE) is an umbrella term coined by Harman and Jones [7] to describe the application of metaheuristic search techniques to problems in software engineering. The branch of SBSE that focuses on software testing is known as search-based software testing (SBST). The application of SBST has been discussed in detail by McMinn [9] for functional, structural, and temporal aspects of testing, and by Afzal et al. [8] for non-functional testing.

Efforts to validate SBST with industrial code do exist. Notable is Fraser and Arcuri's EvoSuite [63], a tool that aims to generate test cases for Java code. The tool has received considerable evaluation, by Fraser and Arcuri [14] on both open source code and by Campos et al. [15] on industrial code. Doganay et al. [16] conduct an evaluation of a hill climbing algorithm on industrial code
derived from Function Block Diagrams developed by their industrial partners. Enoiu et al. [47] conducted an experimental evaluation, also on industrial code, and with master students as experimental subjects.

All these evaluations are conducted by researchers on open source or industrial code, and there is little discussion of transferring the tools used to practitioners. Such a transfer, even it its initial stages, has the potential of showing problems that have thus far been ignored and further avenues for improvement. An evaluation by Fraser and Arcuri on the difficulties encountered in applying EvoSuite “in the real world” [45] discusses the fragility of research prototypes and mentions that even EvoSuite was lacking essential functionality that would allow it to work “on real code”. That study identifies a number of challenges and classifies them into the Usability (e.g. readability of the resulting test cases), Engineering (e.g. integrating with the environment), and Research (e.g. data collection) categories.

The assessment of SBST on industrial code is an essential first step towards transferring this technique to industry. In spite of their rigor and depth, however, these studies do not show a complete picture of how SBST could be transferred to industry. The tools developed and presented are often used by researchers and students, rather than industrial practitioners, and the evaluations are conducted on “historical” code, rather than living projects that are still in development. The issue of how these findings, tools, and techniques can be transferred to industry is seldom discussed.

Vos et al. [18] also discuss the use of evolutionary techniques for black box testing in an industrial setting. In addition, the transfer of the technique to industry is also actively discussed and considered. The authors conclude that the technique was successful, that evolutionary functional testing is “both scalable and applicable”. Nevertheless, they concluded that “a certain level of evolutionary computation skill” is necessary to allow prospective users to define and refine a suitable fitness function, and that the process of defining the fitness function is time consuming. Thus, transfer to industry would depend on ensuring that prospective users have such skill, or can be supported by researchers. This difficulty in defining a fitness function, together with the need for guidelines and benchmarks, are identified as significant factors preventing more widespread use of evolutionary testing in industry.

The interaction between search-based systems and their users has also been explored. Users of search based systems can define specifications [27], or interact indirectly [12, 59]. A more direct type of interaction involves the user directly in the process of assessing solutions that a search-based system finds. For example, Takagi defined Interactive Evolutionary Computation to allow the user to guide
a search-based system according to their “preference, intuition, emotion and psychological aspects” [10], while Tonella et al. [64] proposed a system that allowed the user to intervene to break ties in fitness scores. Other approaches involve adapting the fitness calculation to account for user preference [68, 69], to include elegance [70], or to ensure that candidates that are known to be good receive a higher fitness score [67]. Existing work focuses on interaction with users, but often this interaction is assessed in isolation. In industry, the interaction between the user and an SBST system takes place in the wider context of the organization’s software development and testing processes. The exact interaction between the user and a search-based system is contingent on many factors, e.g. the intended users, the intended goal of the application, the context.

It is also relevant to discuss existing work on the transfer of technology to industry. Gorschek et al. [48] present a technology transfer model that seeks to assess how a research result can move from academia to industry. They describe a number of steps, going from evaluation in academia, static evaluation, and dynamic evaluation in industry. This work provide a useful lens through which the maturity of existing SBST systems can be assessed, and missing elements can be identified.

8.3 Context and Artifacts

8.3.1 The Technology Transfer Model

The Technology Transfer Model (TTM) proposed by Gorschek et al. [48], describes seven steps that technology transfer project go through, along with guidance about putting each of the steps into practice. The TTM steps are:

1. Problem Identification. This step focuses on understanding the context of the industrial partner that will be the beneficiary of the technology transfer project. Understanding the domain, establishing a common terminology, understanding and prioritizing the needs of the industrial partner are identified as key issues at this step.

2. Formulate a research agenda. Based on the needs identified and prioritized at the previous step, researchers formulate an agenda for their work, in close cooperation with their industry contacts.

3. Formulate a candidate solution. A candidate solution is developed for the context, or adapted to fit the context.
4. Validation in Academia. Once the solution is developed, it is validated in a laboratory setting.

5. Static Validation. Static validation consists in having practitioners evaluate the candidate solution, providing feedback to further improve the candidate solution. This type of evaluation takes place in an industrial setting and uses industrial artifacts, but is not carried out in an active project.

6. Dynamic Validation. Dynamic validation consists in evaluating the candidate solution as a pilot in industry. This step is aimed at further improving the solution and indicating what is needed for the full scale transfer. The dynamic validation is carried out as part of an active pilot project.

7. Release the Solution. This step involves delivery of the candidate solution to industry, along with documentation and reference guides, training support, and measurement programs.

The model identifies a number of key issues for the successful transfer of technology to industry. First is the matter of identifying the context and understanding the needs of the industrial partner. Second, the importance of adapting a candidate solution to the context, of tailoring the solution to fit the problem and the company. Finally, the model describes an iterative approach to validation, with the candidate solution being validated first in an academic setting, then being subjected to a static validation on historical data, and then a dynamic validation, in active projects. In addition to increasing the realism of each validation, the model argues that additional information emerging from these evaluations could lead to further modifications and improvements to the candidate solution. Thus, each validation step can lead to a re-appraisal of the candidate solution, and can lead to improvements being made. The updated candidate solution is then subjected to the same set of validations, until it is ready for deployment.

The TTM forms a useful framework for discussing the transfer of SBST in an industrial setting. The evolution of the candidate solution we adopted for the transfer of ISBST in industry will be discussed in the following sections.

8.3.2 Industrial Context

Our industrial partner is a company offering hardware and software products for off-highway vehicles, as well as components for those products. In addition
to developing and testing embedded software themselves, the company offers an embedded software development environment that allows customers to modify embedded software and develop their own modules. Customers use existing modules and components to build function block diagrams (FBD) with the intended functionality. The diagrams are then translated to code, compiled, and deployed on hardware components.

The context of our industrial partner, and of their customers, places a premium on domain knowledge, rather than knowledge of software development techniques and approaches. It also emphasizes quality of the software and hardware components, but without making software central to the company’s business model. As a result, a lot of the developers are specialized in their respective domains, with software development being an important, but secondary, part of their work. We will refer to them as “domain specialists” rather than software developers, to emphasize this focus. The company wishes to enhance the software development environment to support the domain specialists in developing and running test cases.
8.3 Context and Artifacts

8.3.3 The ISBST system

The ISBST tool is a search-based software testing tool was developed to allow domain specialists to use their knowledge and experience to guide the search. This guidance is achieved by allowing the domain specialist to change the fitness function guiding the search, and then assess the resulting test cases to further improve their definition of the fitness function. The fitness function is composed of a number of criteria, called search objectives, that measure characteristics of the output or input of the SUT. The domain specialist guides the search by deciding on the relative importance of these objectives.

The ISBST system has two nested components: an SBST system that connects to the SUT forms the inner cycle, and the outer cycle that handles the interaction between the inner SBST system and the domain specialist. An overview of the ISBST system can be seen in Figure 8.2.

The Inner Cycle consists of the search algorithm itself, the fitness function and the search objectives that form it, and the mechanism that handle the interaction with the SUT. The algorithm used is a differential evolution algorithm [49] that generates a set of 50 test inputs, that are then used to run the SUT and obtain the corresponding behavior. Each test input consists of a vector of real numbers. The combination of inputs and behavior are referred to
collectively as a candidate. Once the behavior has been recorded, the candidate is assessed using the fitness function.

The mutation strategy the ISBST system uses to develop new candidates is as follows:

\[ v_{j,G+1} = x_{r_1,G} + F \times (x_{r_2,G} - x_{r_3,G}) \] (8.1)

where \( r_1, r_2, r_3 \in 1, 2, \ldots, NP \), are integers, and mutually different, and different from the index \( j \) of the new candidate. \( NP \) is the total number of candidate solutions, and \( G \) is the number of the current generation. \( F \) is a real and constant factor \( \in (0, 2] \) which controls the amplification of the differential variation \( (x_{r_2,G} - x_{r_3,G}) \). If the mutant vector is an improvement over the target vector, it replaces it in the following generation [49].

The crossover rate we used is \( cr = 0.5 \), the scale factor is \( F = 0.7 \), and the population size is 100. The mutation strategy is that proposed by Storn and Price [49]: DE/rand/1/bin. The strategy uses a differential evolution algorithm (DE); the vector to be mutated is randomly chosen (rand); one difference vector is used (1); the crossover scheme is binomial (bin).

The fitness function is made up of several search objectives assessed independently. The results of each of these assessments are collected and combined according to Bentley’s Sum of Weighted Global Ratios [50], as can be seen below:

\[ DFF_j = \sum_{i=1}^{nObjectives} \text{Weight}_i \times \text{Value}_{i,j} \] (8.2)

where \( DFF_j \) (the Dynamic Fitness Function) is the fitness value of candidate \( j \), \( \text{Weight}_i \) is the current weight of the objective \( i \), and \( \text{Value}_{i,j} \) is the fitness value of candidate \( j \) measured by objective \( i \). The value of \( DAFF_j \) is the sum of the weighted fitness values for all \( nObjectives \) objectives. An objective \( k \) can be deselected from the computation by having \( \text{Weight}_k = 0 \).

The Outer Cycle is a shell around the SBST system that allows domain specialists to interact with the SBST by adjusting the relative importance of each search objective and to view the resulting candidates. The candidates resulting from the search are displayed as a group, relative to the fitness values they received. Each individual candidate can be displayed in more detail, if a domain specialist deems it useful. The search interaction is conducted by allowing the domain specialist to set the relative weights for each search objective. The weights are then passed to the Inner Cycle, where they form a part of the fitness evaluation.

Candidate solutions are displayed, and interaction is permitted after a fixed number of iterations of the Inner Cycle. For the system presented and evaluated
in this paper, interaction was set to take place every $n_{\text{iterations}} = 50$ iterations of the Inner Cycle.

At the moment, new search objectives can only be added by hand, with the code for the fitness evaluation being added to the appropriate module. Once the code is written, however, the new search objectives are automatically used for future fitness evaluations. However, experience has shown that any set of search objective that is pre-defined is unlikely to be complete, so a means of allowing new objectives to be added would be useful for practical deployment and further evaluation.

### 8.3.4 The development and previous evaluations of the ISBST system

In addition to hardware and software, our industrial partner provides their customers with a development environment that allows customers to modify and develop embedded software. The project to transfer SBST to industry was based on the need of our industrial partner to enhance their development environment to also provide support with developing test cases for the embedded modules being developed.

The flexibility of SBST, along with the capabilities exhibited in other domains, make SBST a good candidate for providing testing support for a wide variety of modules. Thus, SBST was chosen as the underlying mechanism for test case generation. The prospective users would be domain specialists, so we decided to encapsulate the SBST component to allow a user to guide the search without requiring them to become specialists in search-based techniques. The first two steps of the TTM, the problem identification and research problem formulation, were carried out iteratively, using the problem formulation to validate and improve our understanding of the research problem, and allowing this improved understanding to refine the research agenda.

The candidate solution we envisioned was an Interactive Search-Based Software Testing (ISBST) system. We decided to develop an interaction component, that would allow the domain specialist to contribute their domain knowledge and experience to the search. Thus, the domain specialists would have an intuitive interface to define the direction of the search without the need to become experts in search-based software testing. An initial design for the ISBST system was proposed [60]. In the context of the TTM, the formulation of the candidate solution is defined as a single step, but in practice, the candidate solution is redefined and updated as more information becomes available from the valida-
tions in academia and industry. An overview of the latest version of the ISBST system can be seen in Section 8.3.3.

The validation in academia and static validation in industry proceeded simultaneously, focusing on different aspects of the ISBST system. An initial evaluation of the mechanism chosen for guiding the search was conducted in academia [94], and a validation of the visualization component was focused on industry practitioners [95]. This information allowed us to update the ISBST system prototype, and conduct a static validation [73] of the ISBST system in an industrial setting and with industry practitioners.

The evaluation in industry validated our choice of interaction mechanism and of the general concept of the ISBST system. As stated in the Technology Transfer Model, the purpose of the static validation is to get feedback and ideas for improvements, validating understanding, and giving feedback to the practitioners involved in the assessment phase in the previous steps. Based on the feedback obtained, the ISBST system was updated to improve performance and accessibility. The updated ISBST system uses the executable modules, rather than the manually instrumented code required by the previous version. This means that new modules can just be plugged in, without any additional effort, and the module being tested is the compiled version that would be deployed on hardware. In addition to improvements to the ISBST system, the evaluation methods used were also reviewed and improved. In particular, we identified potential search strategies used by the industry practitioners, and incorporated those strategies into follow-up evaluations in academia.

The changes to the ISBST system required us to go through the validation in the laboratory and static validation steps again. The additional validations used lessons learned from previous versions and focused on the effect of interaction on the search process [92], and on investigating the use of exploration to augment the ISBST system [93].

These efforts, however, validate the updated ISBST system in the laboratory, in an academic setting. Before moving towards deploying the ISBST system, a second iteration of the static validation step is required. The new static validation would use the results of previous evaluations, in industry and academia, to refined the objectives of the evaluation, in addition to using an updated system.

### 8.3.5 System under Test

For the purpose of this evaluation, the SUT used was a Time Ramp module, part of the standard library of modules provided by our industrial partner. The module is often used as a component in function block diagrams (FBD) that
### Table 8.1: The input and output signals of the SUT used for the evaluation. The variable types are: U16 - unsigned 16-bit integer; S16 - signed 16-bit integer; BOOL - boolean value.

<table>
<thead>
<tr>
<th>Signal</th>
<th>Values</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input_0</td>
<td>Loop Tm</td>
<td>U16 Processing time of one program loop (Range: 0 to 65535; Unit: 1ms)</td>
</tr>
<tr>
<td>Input_1</td>
<td>Reset</td>
<td>BOOL Sets output equal to Reset Val. (T: Use to preload output at startup to non-zero value. F: Output follows other conditions normally.)</td>
</tr>
<tr>
<td>Input_2</td>
<td>ResetVal</td>
<td>S16 The value set as the output when Reset is true. (Range: -32768 to 32767)</td>
</tr>
<tr>
<td>Input_3</td>
<td>Range</td>
<td>U16 The step size of change in output over DecTm or IncTm (Range: 0 to 65535)</td>
</tr>
<tr>
<td>Input_4</td>
<td>DecTm</td>
<td>U16 The time it takes for the output to decrease in magnitude from Range to 0 (if DecTm = 0, decreasing ramp function is disabled. If 0 &lt; DecTm &lt; LoopTm, LoopTm is used as DecTm; Range: 0 to 65535)</td>
</tr>
<tr>
<td>Input_5</td>
<td>IncTm</td>
<td>U16 The time it takes for the output to increase in magnitude from 0 to Range (if IncTm = 0, decreasing ramp function is disabled. If 0 &lt; IncTm &lt; LoopTm, LoopTm is used as IncTm; Range: 0 to 65535)</td>
</tr>
<tr>
<td>Input_6</td>
<td>Input</td>
<td>S16 Input Signal to be ramped. (Range: -32768 to 32767)</td>
</tr>
<tr>
<td>Output_7</td>
<td>Dec</td>
<td>BOOL Decreasing. (T: The Output is changed by the whole amount of Decrease Time toward zero)</td>
</tr>
<tr>
<td>Output_8</td>
<td>Pasv</td>
<td>BOOL Passive (T: Output is currently unchanged)</td>
</tr>
<tr>
<td>Output_9</td>
<td>Output</td>
<td>S16 Ramped version of the Input signal. Will be equal to ResetVal if Reset is TRUE. (Range: -32768 to 32767)</td>
</tr>
</tbody>
</table>

describe other software modules. This function block provides a timed transition from one value to another, with additional features such as signal reset. Input data types must exactly match the types indicated in Table 8.1.

For this system, a number of search objectives were developed in collaboration with industry practitioners. These search objectives can be seen in Table 8.2.

It is worth mentioning that the module used for the evaluation discussed in this study, like the modules used in previous evaluation of the ISBST system, were already in production at the time of the evaluations and were included in the standard library that our industrial partner and their customers use on a regular basis. As a result, those systems had already undergone rigorous testing and have been used extensively. Therefore, we do not expect that more testing will reveal additional faults in these modules.
### Search Objective

<table>
<thead>
<tr>
<th>Search Objective</th>
<th>Tag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Minimize Output Minimum</td>
<td>minimum</td>
<td>The minimum value of the output signals is computed. Smaller values have better fitness. Since Output_9 is S16, and other output signals are Boolean, this value refers to Output_9. For multiple output signals, this refers to the minimum value of all signals. A similar objective can be developed for each individual signal.</td>
</tr>
<tr>
<td>Maximize Output Maximum</td>
<td>maximum</td>
<td>The maximum value of the output signals is computed. Higher values have better fitness. Since Output_9 is S16, and other output signals are Boolean, this value refers to Output_9. For multiple output signals, this refers to the maximum value of all signals. A similar objective can be developed for each individual signal.</td>
</tr>
<tr>
<td>Output Signal Amplitude</td>
<td>amplitude</td>
<td>The difference between the minimum value and the maximum value of a given signal. Higher amplitudes have better fitness. The objective refers to Output_9. In the case of multiple output values, the one with the higher amplitude gives the fitness value. Individual versions of the objective can be developed for each signal.</td>
</tr>
<tr>
<td>Maximize Output Signal Increase</td>
<td>max. increase</td>
<td>Measures the highest increase in the values between consecutive points of a given output signal. Higher increases give better fitness values. In this example, this refers to Output_9. For multiple output signals of comparable type, the highest increase found gives the fitness value. Individual versions of this objective can be developed for particular output signals.</td>
</tr>
<tr>
<td>Maximize Output Signal Derivative</td>
<td>max. derivative</td>
<td>Calculates the derivative of a given output signal. Higher values of the derivative give better fitness values. In this example, this refers to Output_9. For multiple output signals of comparable type, the highest increase found gives the fitness value. Individual versions of this objective can be developed for particular output signals.</td>
</tr>
<tr>
<td>Minimize Output Signal Mean</td>
<td>min. mean</td>
<td>Calculates the mean of a given output signal. Lower values of the mean give better fitness values. In this example, this refers to Output_9. For multiple output signals of comparable type, the lowest mean found gives the fitness value. Individual versions of this objective can be developed for particular output signals.</td>
</tr>
<tr>
<td>Maximize Output Signal Decrease</td>
<td>max. decrease</td>
<td>Measures the highest decrease in the values between consecutive points of a given output signal. Higher decreases give better fitness values. In this example, this refers to Output_9. For multiple output signals of comparable type, the highest decrease found gives the fitness value. Individual versions of this objective can be developed for particular output signals.</td>
</tr>
</tbody>
</table>

Table 8.2: The search objectives and their definition
The module was chosen for the study since it is a typical software module for our industrial partner. Its inclusion in the standard library and its use on a regular basis also point to a highly relevant and widely used piece of software.

8.4 Static validation of the latest ISBST update

In the previous section, we discussed the history of evaluation and update that the ISBST system received. As a result of the updates, and in line with the recommendations of the TTM, we ran a second round of laboratory evaluations and validation. The additional evaluations showed that the ISBST system had improved, but further evaluation and validation is still required. This section will describe this round of static validation, highlighting the differences and updates in terms of the ISBST system itself, as well as in terms of the evaluation and validation methods used.

8.4.1 Research Questions

Previous evaluations have focused on the domain specialists’ evaluation of the usefulness and usability of the ISBST system, and assessing the effectiveness of the interaction between domain specialists and the ISBST system. As a result of lessons learned in that evaluation, the research questions have been updated. The current research questions make a distinction between the ability of the ISBST system to develop interesting test cases and how clearly the findings of the ISBST system are communicated to the domain specialists.

The study presented in this paper focuses on the following research questions:

1. Does the ISBST system develop test cases that can identify bugs in the SUT? We consider that a set of test cases identifies a bug if it causes the SUT versions with the said bug to behave differently from the reference, bug-free, version.

2. To what extent can domain specialists, using the ISBST system, develop test cases that identify the bugs in the SUT? We consider that test cases developed by the domain specialists using the ISBST system identify a bug if that population of test cases causes the SUT versions with bugs to behave differently from the reference SUT version.

3. To what extent does the ISBST system communicate its findings to the domain specialists? Once the ISBST system has developed test cases that
can identify a bug in the SUT, can domain specialists clearly identify those test cases as exhibiting interesting or incorrect behaviors?

The opinions, comments, and feedback of the domain specialists, as well as their subjective assessment of the ISBST system are still of interest, of course. However, the current study focuses more on the ability of the domain specialists to use the ISBST system, to provide guidance for the search that allows the system to develop interesting test cases, and on the ability of the ISBST system to communicate its findings clearly.

8.4.2 Method

To answer the first research question, a laboratory experiment was conducted. The experiment used the SUT selected and described in Section 8.3.5, and the latest updated version of the ISBST system described in Section 8.3.3. The design of the experiment was further improved on the basis of information obtained in previous evaluations regarding the performance and interface of the ISBST system, as well as our improved understanding of the way domain specialists interacted with the ISBST system in previous evaluations.

The selected SUT is part of a library of modules that have been in use for some time. As a result, the code in question had been thoroughly tested. For the purpose of this validation, we injected 15 faults, creating 15 additional SUT versions with bugs. The injected faults were based on existing work that focused on commonly occurring types of faults in this type of system [96, 97], with three bugs injected for each category. The exact faults that were injected cannot be discussed in detail, due to the proprietary nature of the code, but the categories of these faults are discussed below.

The categories of faults are: 1) CVR (Constant Value Replacement); 2) IID (Inverter Insertion or Deletion); 3) ABR (Arithmetic Block Replacement); 4) CBR (Comparison Block Replacement); 5) LBR (Logical Block Replacement).

To reduce the chance that interactions between different bugs would bias the assessment, a separate SUT version was developed for each of the injected bugs. The ISBST system was used on each of the SUT versions, both with and without the injected bugs, and developed a set of test cases. This set of test cases characterized the behavior of that SUT. The behaviors of the bug-injected SUTs were compared against the behavior of the reference, i.e. bug-free, original SUT.

Laboratory experiments. For the laboratory experiments, the ISBST system was run on each SUT for the same number of interaction events. For each
interaction event, the number of fitness evaluations is the same. The number of fitness evaluations is the main metric for evaluating the amount of effort expended by the ISBST system, based on the work of Črepinšek et al. [83]. For each SUT the system was run for 10 interaction events, with $n_{steps} = 50$ optimization steps between interaction events, resulting in a total of $n_{evaluations} = 500$ evaluations of the fitness function for each SUT version.

We deemed that the bug injected in a particular SUT version was found if the behavior of that SUT was significantly different from that of the reference, bug-free, versions. The comparison was done based on the search objectives, as well as other metrics, discussed below. The difference was significant if, for at least one of the search objectives, and one of the additional metrics, there was a statistically significant difference between behaviors.

**On-site evaluation.** To answer the remaining research questions, an on-site evaluation was conducted with domain specialists from our industrial partner as participants. The evaluation was based on a subset of 6 SUT versions, the bug free version used as reference, and one version representing each of the injected fault categories. The evaluation was conducted with three domain specialists from our industrial partner. They were all practitioners that had not been directly involved in the development or previous evaluations of the ISBST system.

The participants were provided with a brief introduction, to familiarize themselves with the ISBST system, the information it provided, and the mechanism for guiding the search. The introduction was a hands-on experience, where the participants ran the ISBST system on the bug-free version. After this introduction, participants evaluated each of the subsequent 5 SUT versions with injected bugs. Participants were allowed as much time as they needed to complete their assessment, and each participant’s evaluation lasted 1 – 2 hours. The participants were accompanied by a researcher, to provide answers to questions and to record their feedback and comments.

A lightweight version of the Think Aloud protocol was used to explore the participants’ thinking, interpretation of the available data, and to identify any information that is missing, misleading or misinterpreted.

**Assessing behavior differences.** We determine the ISBST system to be successful at finding faults if the population of test cases it produces cause the SUT variants containing faults to behave differently from the bug-free reference version. To determine if a different behavior was observed we use two sets of criteria. The first set of criteria is constituted of the search objectives that are included in the ISBST system and are described in Section 8.3.3.
In addition to the search objectives, we also developed a number of additional metrics to compare the behaviors of different SUT versions. The additional metrics have been used for subsequent analysis, but were not shown to the domain specialists and did not have an impact on the search process. These metrics can be seen in Table 8.3, and have been developed to validate the ISBST system and our previous assumptions:

- The objectives that guided the search were developed and selected after discussions with domain specialists, and validated in industry and in academia. Nevertheless, the possibility exists that the behaviors of the SUTs were not completely captured by these objectives. So an additional set of relevant metrics was selected, to further validate the search objectives and provide a better understanding of the SUT behaviors.

- To test the potential for such measurements in future versions of the ISBST system. The current set of search objectives focus on extreme values in the output signals and on the variation in the output signals. One potential avenue of future improvement for the ISBST system is the development of additional search objectives, using more detailed metrics. One such idea is to measure the distance between input and output signals, and to find test cases where a large discrepancy exists between input variation and output variation. For distance measurements between Boolean signals we used the Longest Common Subsequence, and as a distance measurement between numeric signals we used the Euclidean Distance and the SAX distance [98]. An additional measurement between a current version and a reference population, using Mahalanobis distance, could also be useful for regression testing.

- To illustrate the importance of domain knowledge and SBST knowledge. The measurements compare specific signals based on the assumption that a connection between them is indicative of correct or incorrect behavior. This assumption is based in the detailed knowledge of the particular SUT being tested. Such information is not available to us when developing a general software testing tool, but it is available to the domain specialist, when applying the tool. An example is the Longest Common Subsequence 1-8. The domain knowledge component is that Output_8 expresses whether the output signal is passive. It shows true in two circumstances: if the previous value of the output signal is equal to the current value, and if the reset signal has been triggered. The SBST knowledge part is that, given the current search algorithm and input value generation, it
### 8.4 Static validation of the latest ISBST update

<table>
<thead>
<tr>
<th>Additional Metric</th>
<th>Tag</th>
<th>Definition</th>
</tr>
</thead>
<tbody>
<tr>
<td>Longest Common Subsequence 17</td>
<td>LCS 17</td>
<td>Longest Common subsequence between signals Input _1 and Output _7.</td>
</tr>
<tr>
<td>Longest Common Subsequence 18</td>
<td>LCS 18</td>
<td>Longest Common subsequence between signals Input _1 and Output _8.</td>
</tr>
<tr>
<td>Euclidean Distance 29</td>
<td>E 29</td>
<td>The Euclidean distance between Input _2 (the reset value signal) and Output _9 (the output signal). If the reset value is triggered often, the distance between the two signals should decrease.</td>
</tr>
<tr>
<td>Euclidean Distance 69</td>
<td>E 69</td>
<td>The Euclidean distance between Input _6 (the signal to be ramped) and Output _9 (the output signal). If the reset value is triggered often, the distance between the two signals should increase.</td>
</tr>
<tr>
<td>SAX Distance 29</td>
<td>SAX 29</td>
<td>The SAX distance between Input _2 (the reset value signal) and Output _9 (the output signal). If the reset value is triggered often, the distance between the two signals should decrease.</td>
</tr>
<tr>
<td>SAX Distance 69</td>
<td>SAX 69</td>
<td>The SAX distance between Input _6 (the signal to be ramped) and Output _9 (the output signal). If the reset value is triggered often, the distance between the two signals should increase.</td>
</tr>
<tr>
<td>Mahalanobis distance to reference</td>
<td>M-ref</td>
<td>The Mahalanobis distance from the value of Output _9 (the output signal) for the current version to the same signal of the reference (i.e. bug-free) version.</td>
</tr>
</tbody>
</table>

Table 8.3: The additional measurements included for the analysis.

is unlikely for the input signal to be stable and result in a stable output signal. This would mean that Output \_8 would be true only when the reset signal, i.e. Input \_2, is true.

The additional measurements were not presented to any of the domain specialists during the evaluation process, and were applied after the assessments had already been completed. Thus, the additional measurements only used as an analysis tool. The additional metrics are a diverse set of distances between different signals of the same candidate, or the distance between a certain signal of the candidate compared to the same signal observed in the reference version. A diverse set of distances was used, to ensure a robust evaluation. In addition to the Euclidean distance we also used Symbolic Aggregate approXimation (SAX) Distance [98], Longest Common Subsequence [99], and the Mahalanobis Distance.

SAX [98] is a symbolic representation of time series that allows a time series of arbitrary length \( n \) to be reduced to a string of arbitrary length \( w \), typically
with \( w \ll n \). The algorithm turns a continuous time series in a discrete symbolic representation, that allows the use of existing data-structures and string-manipulation algorithms in computer science. A distance measure can also be defined on this representation. The software developed by our industrial partner and their customers commonly uses time series as input and output signals, so the ability to have a discrete representation for a time series of arbitrary length, as well as a distance defined on that representation, is a useful addition to the set of existing tools. While the input and output signals used in this evaluation are limited to a set number of discrete values, use of SAX as a representation for such signals would allow the distance to be extended to longer input or output signals.

Longest Common Subsequence \([99]\) is a way to compare two strings and determine the maximal common subsequence. In our case, domain knowledge provided the impetus for this assessment. For the SUT used in this evaluation, one of the input signals and one of the output signals were known to be equal, under ideal circumstances. While this measure cannot be generalized to other SUTs, it provides a good example of a relatively simple, purpose-build measurement that can highlight obvious faults. When developing the system, we observed that discrepancies between signals that were meant to be identical were easy to identify as faulty, but difficult to observe in the large amount of information being provided and difficult to communicate to prospective users.

Mahalanobis distance \([100]\) is a measure of the distance between a point \( P \) and a distribution \( D \), introduced by Mahalanobis in 1936. Mahalanobis distance accounts for covariance between variables, when calculating the distance. In addition, Mahalanobis distance is less sensitive to scale differences between variable values. Thus, variables that are correlated, or that are expressed as higher values, do not unfairly influence the distance measurement.

### 8.4.3 Results and Analysis

**The laboratory experiment**

We consider that the ISBST system has “found” a bug if the behavior observed for the version with the injected bug differs significantly from that of the reference, bug-free, version. Note that this evaluation is focused on the underlying algorithm, and provides little information about the interaction and information communication component of the ISBST system. Assessing how useful or intuitive the interaction is, or how usable the system and how well it integrates with existing tools and processes, could not be done in any meaningful way in academia.
Table 8.4: Objectives that show significant differences between SUT versions with injected bugs and the reference version

Table 8.4 shows the SUT versions that exhibit significantly different behaviors from the reference version, and the objectives that identify those differences. We define significantly different behaviors to be behaviors for which the scores for at least one of the search objectives show a statistically significant difference from the reference version. Note that no single objective can identify all the behaviors for systems with injected bugs, but that all the bugs are identified by one objective or a combination of objectives.

Figure 8.3 shows an example of two of the additional metrics that highlight the different behaviors between SUT versions: the Longest Common Subsequence between signals Input_1 and Output_8 on the left and the SAX distance between signals Input_6 and Output_9 on the right. Knowing the characteristics of the SUT, signals Input_1 and Output_8 should be identical, and any set of longest common subsequence values that is below maximum is “unexpected behavior” that should be shown to the domain specialist for evaluation. This is evident for SUT versions 7, 11, and 14. This is one example of how a combination of SBST and domain knowledge can develop a very SUT-specific metric for assessing behaviors. While this metric is not generalizable, similar metrics can be developed for comparing Boolean signals.
Figure 8.3: Overview of two of the additional metrics. Longest Common Subsequence Input 1 and Output 8 on the left, and the SAX distance between signals Input 6 and Output 9 on the right. The X axis shows the respective SUT versions, with 1 being the bug-free reference version. The Y axis shows the values for the respective metrics, normalized as percentages of the minimum and maximum values seen in the dataset.
The second metric, the SAX distance, compares Input.6, the signal to be ramped, to the output signal Output.9. Distances that are significantly higher than the reference values could mean that the output signal is dissimilar enough from the input not be suitable as a ramped version. Distances that are significantly lower than reference values could mean that the signal changes abruptly, which is what this module seeks to prevent.

Note that in both examples we rely heavily on domain knowledge to interpret the results of these metrics and draw conclusions from them.

Table 8.4 shows how all the SUT versions with injected bugs show significantly different behaviors, as measured by the objectives we proposed in the ISBST system and by the objectives suggested for later analysis. The differences marked were manually identified as potentially meaningful and found to also be statistically significant. This means that the underlying SBST system is able to propose test cases that cause the SUTs to behave differently. We conclude that the ISBST system is able, in theory, to identify the injected bugs by comparing the behaviors of the respective SUTs with that of the reference, i.e. bug-free, version. This validates the underlying search-based algorithms we used, and increases confidence in the selection of the search objectives that we used, under laboratory conditions.

On the basis of the results from the laboratory experiment, we can state that the search algorithm, mutation approaches, and selected search objectives were appropriate: the ISBST system was able to distinguish between the behaviors of the buggy versions and that of the reference version of an industrial SUT, under ideal conditions. Thus, in answer to RQ1 we can conclude that the ISBST system does indeed develop test cases that can identify bugs in the SUT, by showing different behaviors between the buggy and the reference versions.

**The industrial evaluation**

To assess the ability of the ISBST system to detect the injected bugs, we looked at the behavior differences observed between the reference and the buggy versions of the SUT, by comparing the test case populations developed by each participant for each SUT version. This evaluation is a similar evaluation to that conducted during validation in academia, but applied to the behaviors developed by the domain specialists. Since the validation in academia relied on our interaction model of the domain specialists’ interaction with the system, a similar evaluation would show if that model is accurate, or if the assumptions made are correct.

Due to time and resource limitations, a subset of the bug-injected SUT versions was selected for the industrial evaluation. One version was selected for each of the categories of bugs injected, as mentioned in Section 8.4.2. In
Table 8.5: Objectives that show significant differences between SUT versions with injected bugs and the reference version in the industrial evaluation. The numbers indicate which candidate’s data shows significant differences between the injected versions and the reference version.

<table>
<thead>
<tr>
<th>SUT Version</th>
<th>1_v1</th>
<th>2_v4</th>
<th>3_v7</th>
<th>4_v16</th>
<th>5_v11</th>
<th>6_v8</th>
</tr>
</thead>
<tbody>
<tr>
<td>minimum.min</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>2,3</td>
<td>1,2</td>
<td>3</td>
</tr>
<tr>
<td>maximum.max</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td>amplitude</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>max increase</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>max derivative</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1,2</td>
</tr>
<tr>
<td>min mean</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
</tr>
<tr>
<td>max decrease</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
<td>1,2</td>
</tr>
</tbody>
</table>

Table 8.5, we define the selected versions as $i_vj$, where $i$ is the identifier number for the experimental evaluations, and $j$ is the corresponding identifier for the same system in the laboratory experiment.

Table 8.5 shows the significant differences in behavior as identified by the data resulting from the industrial evaluation. The numbers show represent the participant that provided data that showed a significant difference in behavior for the respective SUT version with a given objective. All the participants were able to use the ISBST system to generate test cases, and those test cases did exercise behaviors in the SUT variants injected with bugs that differed from the behavior of the reference variant.

However, Table 8.5 also shows that different participants exercised different behaviors of the SUT versions. A combination of the initial search objectives (above the line) and the supplementary objectives (below the line) shows that the different versions did indeed show different behaviors, and that the injected bugs did manifest in their behavior. However, some of the SUT versions only showed behavior differences, with respect to the initial search objectives, for one of the participants. This suggests that, for some participants, bug-injected
versions were indistinguishable from the bug-free reference version. Additional
search objectives show differences, but the initial objectives can be seen to be
unsuccessful in identifying different behaviors.

Note that the behavior of the search objectives with respect to the SUT
variants is consistent with the behavior observed in the laboratory. Notice that,
not all domain specialists were able to identify differences between the different
versions, in spite of using the same SUT versions and the same ISBST system.
Given the important of personal experience, it is not unexpected to see different
domain specialists using the ISBST system in different ways. However, since
the additional objectives do detect differences between the behaviors observed,
and the included search objectives do not, this suggests the objectives are not
as robust or reliable as needed, but that improvement can be made to them.

The overall conclusion is that validating the objectives in an academic set-
ting, under ideal conditions, does not guarantee that those objectives will be as
effective in findings bugs in an industrial setting. Different domain specialists
may guide search objectives according to different strategies, interpret the find-
ings in different ways, and therefore achieve different outcomes. For example,
participant 1 was able to generate test cases that lead to different behaviors
for some of the SUT versions, but not for others. The additional objectives
performed more consistently, indicating that the SUT versions did indeed result
in different behaviors. In practical terms, test cases that point towards a prob-
lem could be developed, but not recognized as such by the search objectives.
As they would not receive a high fitness value, they may not be shown to the
domain specialists, and may therefore be ignored.

This means that the objectives we have selected for inclusion in the ISBST
system are not always good at indicating faulty behavior to the domain spe-
cialists, in spite of the previous evaluations in workshops and in laboratory
experiments. The differences between expected and observed search objective
performance is difficult to estimate without an evaluation in industry. These
findings suggest that validation in academia is no substitute for an on-site eval-
uation carried out with practitioners. In the short term, the search objectives
that are already identified as suitable can be included in the ISBST system. For
researchers developing search-based systems and seeking to transfer them to in-
dustry, this underlines the importance of extensively validating the mechanisms
chosen for fitness evaluations in their appropriate context.

Therefore, in answer to RQ2, we can state that domain specialists, using the
ISBST system in an industrial setting, were able to develop test case populations
that captured differences between the behaviors of SUT versions with injected
bugs and the reference SUT version. This supports the conclusion that the ISBST system is valid in the context we have evaluated it in.

Communicating search results

The issue communicating the results of the search to domain specialists is a much harder problem to assess. For this evaluation, a researcher was present to make note of the interaction between domain specialists and the ISBST system, as well as to collect impressions, comments, and suggestions given by the domain specialists during the evaluation. Overall, we noted that domain specialists were able to quickly adapt to the interaction mechanisms and were able to use them effectively to guide the search. It is worth mentioning, though, that additional explanations and discussion were necessary regarding the operational details of each of the search objectives.

The information display, however, was more problematic. The general display shows all the current generation of test cases, along with the previous generation of test cases, plotted with respect to their fitness scores. Based on our observations, participants had trouble in discerning what items of the information displayed were relevant. The graphs’ axes were pairs of search objectives, with the participant being able to select which objectives to display. However, this resulted in a large amount of information that had to be accessed in separate graphs. The information was available, however the default visualization did not always include relevant information, or the participants’ attention was not drawn to relevant candidates. As a result, we observed that some relevant information was not noticed, and participants had a hard time identifying problems in the behaviors.

For information on individual test cases, candidates that were deemed to have “interesting” behaviors, were identified as outliers in the general graphs and selected for visualization. Nevertheless, the characteristics that made those candidates outliers, according to the ISBST system, were not communicated well to the domain specialists. As a result, the domain specialists overlooked those characteristics and were unable to identify those behaviors as either correct or incorrect.

As an answer to RQ3, we conclude that the ISBST system is capable of generating test cases that exercise different behaviors for a given SUT under industrial conditions. However, the mechanisms the system uses to display that information to the domain specialists do not seem able to communicate information clearly enough for practical use. In particular, the ISBST system does not clearly express why certain candidates got the fitness scores they did, so domain specialists have a difficult time assessing their behaviors. Further work is necessary to ensure that the ISBST system communicates its findings
in a clear and intuitive way to the domain specialists, and ensures that the reasoning behind the assessment it provides is clear to the domain specialists.

8.4.4 Discussion on the transfer of ISBST to industry

From the perspective of the Technology Transfer Model, the experimental evaluations presented here fit under two steps. The first, the laboratory experiment, falls under validation in academia. It uses industrial code, but is conducted exclusively in the laboratory. It focuses on the ISBST system itself, on the ability of the system to interact with the SUT and create the appropriate test cases. The second experimental evaluation falls under the static validation step. It is conducted on site, with industrial practitioners as participants, and using an industrial SUT. While not part of an active project, the experimental evaluation provides useful feedback about the degree to which the ISBST system could fit in the development environment of our industrial partner.

Previous versions of the ISBST system required source code to be instrumented, to allow test cases to be generated and run. This approach was flawed for two reasons. First, the code being tested did not necessarily behave in the same way as the final product. Since the C code we were testing was further compiled, it could be subjected to optimization that the ISBST system could not account for. There is also the possibility that the instrumentation itself could alter the behavior of the SUT. As a result, the updated ISBST system uses the executable file, the version that is ready for deployment on the hardware, with no need for additional instrumentation or manipulation. We can, therefore, argue that the SUT behavior observed in this evaluation is likely to be closer to behavior in use, and less likely to be influenced by our tools. It is worth pointing out that interaction between hardware and software could also alter the behavior. However, once the system is considered stable enough to be deployed on hardware modules, it is subjected to further testing and quality assessment, and the methods and techniques for that stage are already in place.

The experimental evaluation in industry concluded that further improvements are needed, particularly in the degree to which information is communicated to the domain specialists. Test cases that exhibit extreme behaviors need to be better highlighted, as are the reasons for which those behaviors are considered extreme. It is worth pointing out, however, that the domain specialists adapted quickly to the ISBST system and were able to use it with little intervention from the researcher present.

Based on our experience, solving technical issues, e.g. connecting to the SUT without the need for code instrumentation or manipulation of the artifact, has
proven to be more suitable for an academic environment. Potential solutions to this problem could be developed and evaluated in the laboratory, without requiring external resources. Assessing the clarity of the communication between the ISBST system and the user, however, requires the participation of industry practitioners. Potential solutions for this problem need to strike a balance between assessing as many potentially useful methods as possible, and saturating industry practitioners with evaluations, and risk wasting their time.

8.4.5 Conclusions

As a result of the two experimental evaluations presented above, we conclude that the ISBST system, in its latest iteration, is capable of developing test cases that cause faulty SUTs to exhibit different behavior than the reference, i.e. fault-free, versions. As a result, we conclude that the ISBST system has been validated in academia, and is able to develop relevant test cases in ideal conditions.

The experimental evaluation in industry shows that the ISBST system can work under realistic conditions, and that domain specialists are able to use the system to develop test cases that identify faulty behavior. It also showed, however, that further improvements need to be made, in order to allow the ISBST system to clearly and meaningfully communicate its findings to the domain specialist. Clear and meaningful communication of the result findings would enable domain specialists to more accurately guide the search, but would also allow them to better understand how the ISBST system fits in the company’s quality assurance process, how it interacts with other tools that support that process, and allow them to provide feedback regarding aspects of the system that need improvement.

Within the framework of the TTM, the next step towards technology transfer is dynamic evaluation in an active project. While further work is still needed to prepare the ISBST system for transfer, we conclude that the ISBST system is a viable candidate for transfer.

8.5 Lessons Learned

This section discusses the lessons drawn from the evaluations of the ISBST system presented in this and previous studies, and discusses some of the pitfalls encountered thus far. In the previous and current evaluations, we have gone
8.5 Lessons Learned

through 5 of the 7 steps of the technology transfer model proposed by Gorschek et al. [48].

8.5.1 Lessons on the transfer of SBST to industry

In the following, we will discuss lessons on the transfer of an SBST system to industry, that are based on pitfalls that we encountered or narrowly avoided.

The need for continuous gathering and validation of information throughout the process.

The initial search objective development and selection is based on existing bug databases, and on workshops and interviews with domain specialists. Such initial information may be incomplete, leading to invalid search objectives and approaches, and reducing the relevance of the resulting solution. Thus, we suggest that continuous validation efforts are necessary to ensure that domain knowledge that the objectives are based on is relevant and up to date. The initial efforts to capture relevant information will be incomplete. As more information becomes available, researchers are better able to formulate relevant questions, and domain specialists have a better understanding of what information is necessary, and a clearer set of questions to answer. As a result, we suggest continuously validating the available information, the resulting search objectives, and the selection of search objectives.

Search objective selection.

The first step is aimed at defining a set of search objectives that can detect changes in behavior caused by the existence of bugs, and validating that selection. Search objective selection can be a problem if it is based on incomplete information. Categories of bugs that are not present in the initial information, in the bug databases, or are not mentioned by domain specialists, may be missed by the developers. As a result, the selection of objectives may not be able to detect behaviors that are indicative of those types of bugs.

Bugs that affect the overall behavior of a system.

Certain categories of bugs may change the entire behavior of a particular SUT. For example, replacing a constant value that is used in an additive process might change all the outputs consistently. As a result, specific search objectives may have to be developed specifically for that type of bug. A potential solution could be a comparison between the behavior of the current SUT and some reference set, for example resulting from running the previous versions of the same SUT.

Domain knowledge compromise.
There are two major forces acting on the researchers when developing the search objectives. The first is a desire to minimize the number of search objectives, and to make them as general as possible. This offers benefits in terms of reuse and in terms of generalizability of the results. The second is a desire to incorporate as much domain knowledge as possible, especially when trying to find specific bugs. A compromise is needed to ensure that the search objectives that are used are both useful for the SUT at hand and generalizable.

One example of this emerged during our work applying the ISBST system to the TimeRamp module. One input signal is used to transmit a configuration from previous calculations to the present module. Although the interface is defined as U16, only 3 of the values are meaningful, as they transmit preset information to the module. A SUT-specific search objective could be developed to restrict the search and ensure that computing time, and domain specialist attention, are not wasted on test cases that have meaningless inputs. At the same time, however, this type of search objective can only be used for this SUT. Moreover, generating a large number of SUT-specific search objectives may result in problems with the selection of search objectives appropriate for a given SUT.

We suggest that a compromise can be reached, with search objectives that are generalizable being the main core. Where necessary, flexible categories of objective could be developed, that would allow domain specialists to limit the search on a case by case basis. All these efforts, however, would have to be carefully assessed and validated throughout the process.

Finding a compromise between robustness and early validation.

When evaluating the ISBST system in industry, robustness was a major concern. The system has to be robust enough to be used by domain specialists and should not be prone to random failures or require very specific behaviors from its users. For example, the search process often takes a few seconds. If interaction with the ISBST system during that interval can result in crashes or unpredictable behaviors, this should either be made clear to the user, or interaction should be prevented at that time.

Problems relating to the robustness of research tools has been mentioned before [45], with time and resources being cited as possible causes for this problem. Achieving a compromise between early evaluation of a brittle prototype and late evaluation of a more robust version is a problem that can only be solved on a case by case basis. We suggest that the matter be given active consideration. A brittle prototype may fail to provide the necessary information, and may suggest to industry practitioners that the solution is not ready for transfer. A
robust, but late, version could result in considerable re-work and wasted effort, as additional information becomes available.

**Assessing the suitability of search objectives for industrial use.**

Not all search objectives are suitable for use in an industrial environment. For example, this can be due to brittleness, as discussed above. Another example of this is search objectives that require more time to complete: for the ISBST system, evaluations that take 5-7 minutes were deemed to be too long. Domain specialists using the system became disengaged and found it difficult to use the provided functionality.

Early validation is essential in identifying such search objectives, and in optimizing them to improve execution time, or replacing them with others that offer comparable results.

**The effect of correlation between search objectives.**

We discussed earlier that SUT-specific search objectives may be developed to allow the proposed solution to fully use available domain information. Our evaluation of the ISBST system revealed that this could result in search objectives that are not orthogonal. Correlations between search objectives could adversely affect the search, as the correlated objectives are favored by the same type of behavior and offer higher fitness values than objectives that are not correlated.

Continuous evaluation of the search objective selection would allow researchers to determine if the search is affected by such behaviors, and to correct any problems with the search parameters.

**Tool reliability.**

At this stage, the proposed system is meant to be evaluated in an active project. For this to work, the system has to be reliable enough to use without the constant presence of the researchers and without constant tinkering. At this moment is should be a functional, robust, usable tool. The trade-off we discussed previously, between reliability and early evaluation, stops being an issue. At this stage, the system should have received a significant amount of evaluation and improvement, so resources can be spent on reliability.

**The effect of tool usability on the evaluation.**

In our experience thus far, usability has not been considered a priority. Previous evaluations have been mostly academic, using researchers or automated tools to run the system. For the evaluations that we conducted in industry, a researcher was always present to answer questions, provide information and clarification, and fix any problems with the ISBST tool. Low usability could have a negative impact on tool evaluation, as the efforts and feedback of participants focus more on identifying problems with the tool rather than on assessing
the underlying concept, or the potential uses and problems with the technique. In any tool that should be evaluated in an active project, and later transferred to industry, usability is worth the resource and time investment.

8.5.2 Lessons specific to interactive systems

The lessons above are useful to the transfer of SBST systems to industry in a more general sense. The current and previous studies have also revealed lessons that are applicable in particular to the transfer of interactive SBST systems.

Information overload.

As stated before, a search-based system can generate large amounts of information. The ISBST system displayed a total of 100 test case candidates, comprising the current and previous populations. They could be displayed relative to each other, in a set of 2-dimensional graphs, one for every combination of two search objectives that the domain specialist wanted to visualize. Each selected candidate could also be visualized separately, with the input and output signals displayed on demand.

While all this information was useful, not all of it was equally relevant. For example, identifying outliers with respect to individual search objectives was relatively easy, but outliers with respect to several objectives was not as clear. The relatively large amount of information, as well as difficulty in identifying quickly which items of information were more important or relevant, lead to confusion. Domain specialists were lost in the information provided. We suggest that this can happen even for systems that do not require user interaction, but that do involve people in assessing and interpreting the results.

Early validation could help researchers in identifying this problem. Solutions could vary on a case by case basis. Information could be divided between several different areas of concern, e.g. separating the overall view from the display of individual candidates. Visual aides could be provided: outliers or candidates that the system regards as remarkable in some way could be highlighted, and the reason for this selection provided. Not all of these require significant changes to the functionality of the system, but could make an important difference for any efforts at evaluating, validating, and transferring such systems to industry.

Awareness of the search progress.

Our experience with the ISBST system shows the importance of keeping the user informed of the progress of the search. This will allow users to decide if additional effort spent searching could lead to better results, or if a new approach should be tried.
For example, the system could how the overall fitness values have changed during the search, which search objectives have seen improvements in the fitness scores and which have not. This could be useful in determining if the search is going in a desirable direction.

8.5.3 Overall Lessons

In general, we would like to highlight the importance of early and continuous validation of any tool being transferred to industry. While we assume that any such tools have already been evaluated in academia, they would have to be changed to adapt to the new context, and to fit with the company’s tools and processes. Continuous validation ensures that everything from the search algorithm to the interaction and information display mechanisms are appropriate for the task. We also strongly advise that such evaluations are as close to real operation as possible. This means involving practitioners early, considering tool and process interactions, and validating information display mechanisms. Realistic evaluations also allow practitioners at the company to become familiar with the new tools, allowing them to provide more relevant information and to conduct a better and more informed assessment.

A second general recommendation is to keep tool design flexible. From the initial step, where a search-based solution is proposed and validated in academia, until the final step, when it is ready for deployment, a prototype will undergo significant changes.

Last, we found that communicating information to the industry practitioners is a non-trivial problem. Sufficient information needs to be available to allow practitioners to make an informed decision. That information needs to be presented in a clear and reasonable way, to allow them to quickly understand it and use it for decision making. Domain specific visualization approaches are important, since they are already familiar to potential users and require no additional training to use.

8.6 Threats to validity

This section will discuss the threats to the validity of the entire study, from the initial development of the ISBST system, to the most recent evaluation.

The study is based on our experiences developing and evaluating the ISBST system in industry. The development of the ISBST system was focused on the needs of our industrial partner, and on assessing that system in academia
and industry. While we made every effort to ensure that our conclusions are accurate, some threats to validity still exist, and will be discussed in this section.

We would like to underline that the list of potential problems is not complete or exhaustive. Further efforts will surely reveal additional problems, and ways of addressing them, especially in the dynamic validation phase. The list of lessons presented here should help researchers in initiating projects to transfer search-based technology to industry and in navigating the early phases of such projects.

During our study, the first three steps of the Technology Transfer Model overlapped to some extent. As a result, we will discuss the validity threats relevant for those steps together. The ISBST system was developed based on information from, and to address the needs of, a specific company working with a specific type of embedded software. While we have not yet identified any reason why our conclusion cannot be applied for other types of software systems, or why our lessons are not relevant for other contexts, we cannot safely generalize on the basis of this example alone.

Moreover, the initial information collected from our industrial partner shaped the development of both the ISBST system and of the evaluation mechanisms. Our assumptions were based on the accuracy and completeness of this initial information. In later stages of the project we have made efforts to validate those assumptions, and correct them when they were unsuitable. Nevertheless, it is possible that some of our assumptions are not accurate or generalizable. While we have confidence in our approach and our conclusions, we advise other authors seeking to transfer their research to industry to pay careful attention to their data collection and data validation steps, especially in the initial stages of technology transfer.

We focused on our industrial partner, their context and tool chain, and on the systems they wanted to test and the problems that they were expecting. As a result, the ISBST tool was designed to fit that context and fulfill those requirements. We have made the ISBST tool flexible: it can use different search objectives [73, 92], and different search algorithms [93]. Nevertheless, it is difficult to know how easy it is to transfer the ISBST tool to other domains until such a project is undertaken and the results analyzed.

The following steps, Validation in Academia and Static Validation, will also be discussed together. For these steps, we emphasized the importance of the evaluation mechanisms we developed for the purpose, and on the potential differences between evaluations suitable for academic settings, and those suitable for industrial settings. All the evaluations were developed by researchers. Some of the evaluations, particularly in the Static Validation step, were conducted
by domain specialists, but under the supervision of researchers, and with data collection being conducted by researchers. While we have made efforts to ensure that the evaluations are as objective as possible, there is a possibility that our own biases have influenced the conclusions. As a result, we encourage other researchers to assess our results, conduct similar projects, and share their findings. To this end, we have prepared a replication package, containing the ISBST system, the evaluation mechanisms, and the analysis scripts we used.

An essential issue regarding validation is that of visualization. A good visualization is essential to ensure that participants understand the system, its capabilities, and can provide meaningful feedback. For the ISBST system, we developed visualization tools based on the commonly accepted approach used by the domain specialists, and we continuously improved them throughout the project. While we seem to have achieved our goal to make visualization clear enough to allow good evaluations, we have also identified problems. Further work is needed to fully understand how to develop and evaluation visualization mechanisms, especially visualization mechanisms applicable to a more general type of domain.

The static evaluation described above was conducted in an industrial setting, with industrial practitioners and industrial code. That evaluation has provided evidence that the ISBST system we developed is usable, and that search-based software testing is useful in improving the testing process. However, the evaluation was not conducted in a live project. It was also comparatively short. A dynamic evaluation, conducted in an active project, where the ISBST system is used by domain specialists without researcher involvement, and conducted over a longer time span is needed to confirm the usability and usefulness of search-based software testing in an industrial setting and to provide further feedback.

Lastly, our evaluation was based on a small number of engineers at the company. This may limit the generalizability of our conclusions, especially on subjective considerations like interaction evaluation. The participants in this study were engineers at the company, working with the type of SUT that we evaluated on a daily basis, developing and testing similar systems. We argue that, in spite of their low number, their experience and knowledge makes their evaluation useful and meaningful. Nevertheless, further research is needed before a definitive conclusion can be reached.
8.7 Discussion

The study presented above is based on our experiences developing and implementing an ISBST solution for an industrial context. We used the Technology Transfer Model proposed by Gorschek et al. [48] as a framework to assess the progress of the project to transfer the ISBST to industry.

Existing work argues that search-based techniques cannot be used by domain specialists without the support of experts in evolutionary computation [18]. We acknowledge the difficulties in developing a complex and sophisticated fitness function without experience. By using a set of domain specific search objectives, the ISBST system allows domain specialists to set priorities and guide the search without the need to develop a fitness function by hand. Our evaluations in industry show that this approach is intuitive enough to allow domain specialists to guide the search and develop test cases, even in the absence of experience with search-based systems. The same work, by Vos et al. [18] also finds, however, that evolutionary computation is useful in industry and that the results compensate for the time and effort spent. The authors also identify a number of obstacles that stand in the way of transfer of such techniques to industry. We argue that the ISBST system partially addresses the need for constant support from domain specialists.

We firmly believe that SBSE in general, and SBST in particular, are useful and flexible tool and could provide benefits to industry. The goal of this paper is to promote more applied research into the development, implementation, and application of search-based software systems. More validations in industry would yield additional information about such tools and strengthen confidence in their usefulness.

8.8 Conclusions

Search-based software engineering has received considerable attention from researchers. A lot of the research in SBST is focused on developing new search-based techniques, and evaluating and validating them. Tools such as EvoSuite [45, 63] provide support for SBST research and have received extensive validation on open-source and industrial systems.

In this paper, we presented a project to develop and transfer an Interactive Search-Based Software Testing system to industry. The lessons learned from our own development and evaluation of the ISBST system should prove useful for developing, deploying, and validating such search-based software tools for use
in an industrial context. We encourage researchers to seek early and continuous
interactions with industry, to assess and validate their ideas, and to ensure that
their efforts are relevant and useful for industrial practitioners. We also discuss
the importance of tailoring systems for the benefit of the companies where they
should be used. Interaction is a central concept to ISBST, and may be less
so to other SBST tools. Nevertheless, visualizing the results of a search in an
intuitive and meaningful way, and ensuring that an SBST tool integrates well
with the processes of the company where it will be used are of concern for any
technology transfer.
Bibliography


ABSTRACT

Context. Software is increasingly included as a component of a variety of systems and products across different domains. The ubiquity of software components means that software is often developed for particular domains and products, and its development and quality assurance processes need to fit the context for which they are developed. This means that domain knowledge is crucial in developing, integrating, and ensuring the quality of such software components. In this context, a good software solution would have to be flexible enough to support a variety of software components in a domain, as well as being usable for engineers whose expertise is more focused on their domain than on software development and testing.

Search-Based Software Testing (SBST) promises to be such a flexible approach to generating and evaluating large numbers of test cases at minimal cost. Currently, SBST requires specialized knowledge to appropriately formulate the problem, to implement, and to use, as shown by the general lack of studies showing the application of SBST to industry. The need for specialized knowledge presents an obstacle that prevents more companies from trying, developing, evaluating, and adopting appropriate SBST systems.

Objective. The objective of this thesis is to develop an SBST system that can be applied in industry, evaluate that system in an industrial context, and reflect on the process of transferring this technology to industry.

Method. The thesis combines a series of research methods, both qualitative and quantitative, within the framework of Design Science Research for information systems. A prototype system was developed, evaluated in industry, and updated, as a means to investigate the applicability of the technique and to assess specific problems related to the development and transfer of SBST to industry. Case studies and quasi-experiments were used to assess the prototype in an industrial setting, and controlled experiments were used to further investigate the findings and support our conclusions.

Results. An Interactive Search-Based Software Testing (ISBST) system was developed, evaluated in academia, and validated in industry. In addition, results show the importance of a well thought-out and complete transfer process, from communication with the industrial partner, to the development of appropriate mechanisms to evaluate and validate the ISBST system within the specific industrial context.

Conclusions. We conclude that ISBST can be applied in industry, and that it provides a useful complement to existing software testing techniques. In addition, the process of transferring ISBST to industry is better understood, as are the additional requirements such a transfer places on the proposed solution.