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

Bogdan Marculescu[a], Robert Feldt[a], Richard Torkar[a,b], Simon Poulding[a]

[a] *Blekinge Institute of Technology, Karlskrona, Sweden*
[b] *Chalmers and the University of Gothenburg, Gothenburg, Sweden*

**Abstract**

Search-based software testing promises the ability to generate and evaluate large numbers of test cases at minimal cost. From an industrial perspective, this could enable an increase in product quality without a matching increase in the time and effort required to do so.

Search-based software testing, however, is a set of quite complex techniques and approaches that do not immediately translate into a process for use with most companies.

For example, even if engineers receive the proper education and training in these new approaches, it can be hard to develop a general fitness function that covers all contingencies. Furthermore, in industrial practice, the knowledge and experience of domain specialists are often key for effective testing and thus for the overall quality of the final software system. But it is not clear how such domain expertise can be utilized in a search-based system.

This paper presents an Interactive Search Based Software Testing (ISBST) system designed to operate in an industrial setting and with the explicit aim of requiring only limited expertise in software testing. It uses SBST to search for test cases for an industrial software module, while also allowing domain specialists to use their experience and intuition to interactively guide the search.

In addition to presenting the system, this paper reports on an evaluation of the system in a company developing a framework for embedded software controllers. A sequence of workshops provided regular feedback and validation for the design and improvement of the ISBST system. Once developed, the ISBST system was evaluated by four electrical and system engineers from the company (the 'domain specialists' in this context) used the system to develop test cases for a commonly-used controller module. As well as evaluating the utility of the ISBST system, the study generated interaction data that was used in subsequent laboratory experimentation to validate the underlying search-based algorithm in the presence of realistic, but repeatable, interactions.

The results validate the importance that automated software testing tools in general, and search-based tools, in particular, can leverage input from domain specialists while generating tests. Furthermore, the evaluation highlighted benefits of using such an approach to explore areas that the current testing practices do not cover or cover insufficiently.

*Keywords:* search based software testing, interactive search based software engineering, user centered, embedded software, industrial experience

## 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 [1, 2]. 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 [3, 4, 5, 6, 7], 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 2, we consider existing approaches to interactive evolutionary search, and discuss how our approach differs from them. Section 3 is an overview of the industrial case study, and Section 4 describes the ISBST system developed during the study. Section 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 6. The results of the industrial evaluation and laboratory experiment are discussed in Section 7. Threats to validity are discussed in Section 8. Section 9 concludes the paper.

## 2. Related Work

Search-Based Software Engineering (SBSE) is a term coined by Harman and Jones in 2001 [8] to describe the application of metaheuristic optimization (or 'search') algorithms to software engineering problems, see e.g. [9, 3, 10]. 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 [1, 2], from object-oriented containers [11] to dynamic programming languages [12].

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 [5] 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 [3], and by Parmee et al. [13], 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" [4]. 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" [4]; this includes applications such as arts and animation, computer generated graphics and image processing.

Takagi [4] 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" [14].

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 [15]. One such example is presented by Bavota et al. [16], 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" [7]. 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 [17].

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 [18], 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. [19], and Sun et al. [20] 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. [21], 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 [22] 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. [23], 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 [24], 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 [25], 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 [17]; 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.

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

### *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.

### *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.

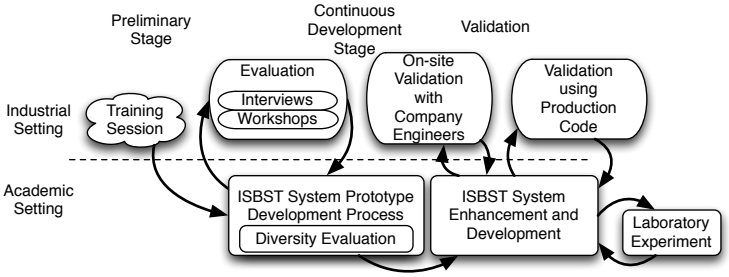



Figure 1: Overview of the Study Design

### *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 [26]. The study design is summarized in Figure 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 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.

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 6.

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

## 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

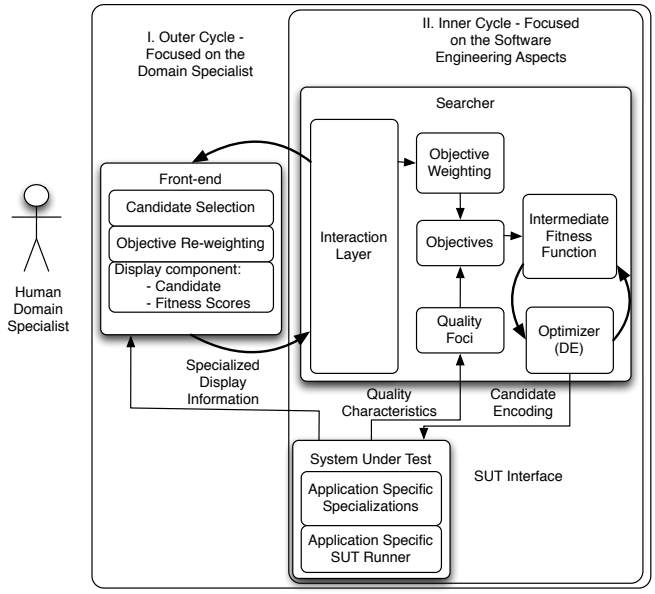



Figure 2: Overview of the ISBST System

above; and the set of quality objectives is specific to the type of software that is developed using the toolkit.

### *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 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 [25], while making allowances for the increasing complexity inevitable in practical implementation.

### *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 1, are calculated from the outputs of the SUT when run with the candidate test case.

| No. | Objective | Description | Aim |
|---|---|---|---|
| 1 | Large Signal at Least Once | Rewards higher maxima values in the output | Maximize |
| 2 | Small Signal at Least Once | Rewards lower minima in the output. | Minimize |
| 3 | Mean Output | Rewards higher mean values in the output. | Maximize |
| 4 | Above Important Limit at Least Once | Rewards test cases with output values that exceed by the most a prescribed maximum value. Only the largest difference is considered. | Maximize |
| 5 | Below Important Limit at Least Once | Rewards test cases with output values that fall farthest below a prescribed minimum value. Only the largest difference is considered. | Maximize |
| 6 | Above Important Limit Overall | Rewards test cases with mean output values that exceed by the most a prescribed maximum value. | Maximize |
| 7 | Below Important Limit Overall | Rewards test cases with mean output values that fall farthest below a prescribed minimum value. | Maximize |
| 8 | One Large Increase | 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. | Maximize |
| 9 | One Large Decrease | 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. | Minimize |
| 10 | Swings Through Zero | 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. | Maximize |
| 11 | Diversity | 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. | Maximize |

Table 1: An overview of the current Quality Objectives

**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 [27]. 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.

$$IFF(j) = \sum_{i=1}^{\text{nObjectives}} \text{Weight}_i * \text{Value}_{i,j} \quad (1)$$

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 $i$. 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 $\text{Weight}_k = 0$.

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 4.3 below.

**Search Algorithm.** The search-based algorithm chosen for this implementation was Differential Evolution (DE) [28].

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:

$$v_{i,G+1} = x_{r_1,G} + F * (x_{r_2,G} - x_{r_3,G}) \quad (2)$$

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 [28].

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 [28]: 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).

### *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 3. To visualize more than a pair of quality objectives, a domain specialist can choose to view a matrix of

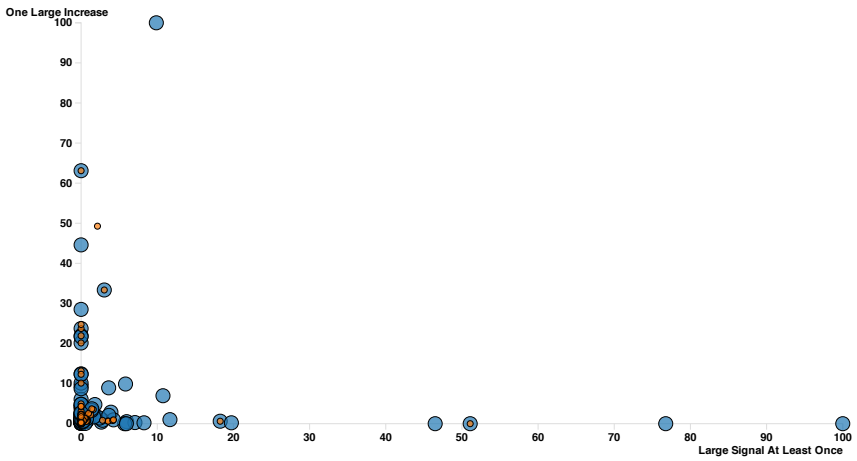



Figure 3: Single Scatterplot example. The current generation is shown in light blue; the previous generation is in orange. (Screenshot from the ISBST tool.)

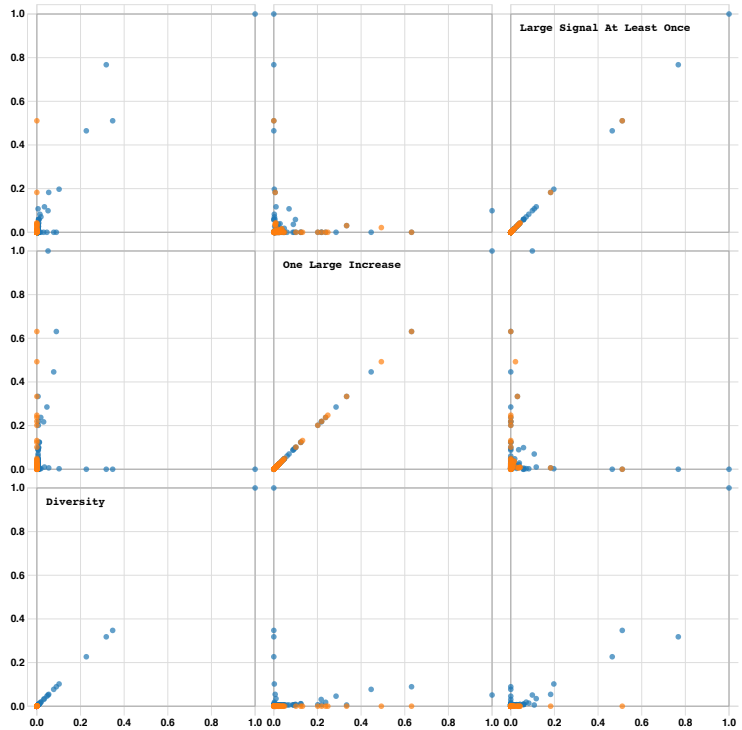
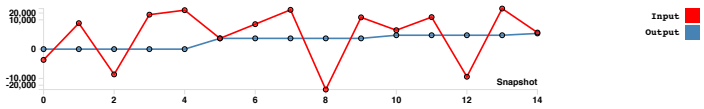
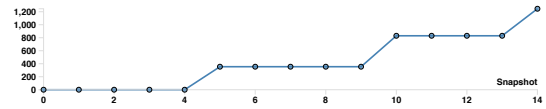



Figure 4: Scatterplot Matrix example. The current generation is shown in light blue; the previous generation is in orange. (Screenshot from the ISBST tool.)




| json_class | id | name | inputs | | | | | snapshots | | | | | | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| RampCandidate | 22618 | Cand84 | sft_strt | sft_end | rnge | decTm | incTm | -354 | 6671 | -6089 | 17153 | 25788 | 354 | 5855 | 26564 | -29124 | 13083 | 2325 | 13410 | -8172 | 29529 | 1498 |
| | | | 59 | 21 | 61132 | 39998 | 3002 | 0 | 0 | 0 | 0 | 0 | 354 | 354 | 354 | 354 | 354 | 830 | 830 | 830 | 830 | 1246 |

Figure 5: A detailed view of one of the test case candidates. (Screenshot from the ISBST tool.)

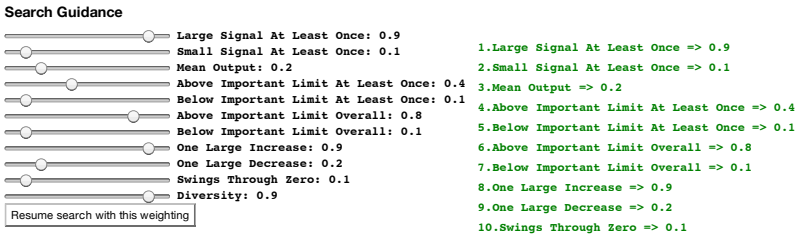



Figure 6: Search Guidance panel. (Screenshot from the ISBST tool.)

plots, where each graph in the matrix shows the fitness values of a pair of the selected quality objectives, e.g. Figure 4.

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.

To guide the search, the domain specialist decides the relative importance of the quality objectives using the Guidance Panel in Figure 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 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.

### *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] 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

[2] `http://coffeescript.org/`
[3] `http://d3js.org/`

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. Industrial Evaluation

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

### *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 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.

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.

[4] `http://www.json.org/`
[5] `http://www.sinatrarb.com/`

| No. | Previous Domain | Experience in the Company (yrs) | Education | Team | Completed Evaluation |
|---|---|---|---|---|---|
| 1 | Control Systems | 14 | Accredited Industrial Electrician | Software | Yes |
| 2 | Electrical Engineering | 2 | MSc. in Electrical Engineering | Software | Yes |
| 3 | Electrical Engineering | 15 | Accredited Industrial Electrician | Software | No |
| 4 | Software Development | 9 | MSc. in Electrical Engineering | DomainDevEnvironment | Yes |
| 5 | Telecom | 19 | BSc. in Computer Science | DomainDevEnvironment | Yes |

Table 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.

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.

### *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

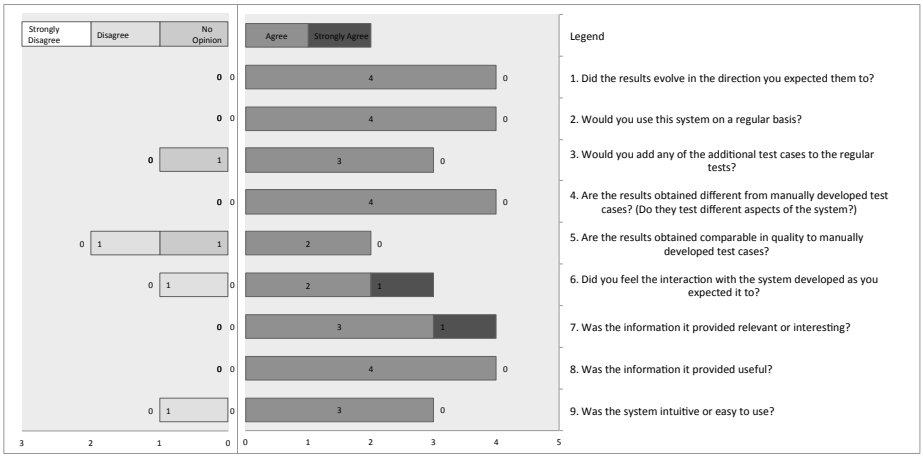



Figure 7: Answers to Final Questions.

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 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.

### *5.3. 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.

## 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.

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

### *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 of Figure 8a 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 would.

| No. | Strategy | Description | Comments |
|---|---|---|---|
| 1 | Null Strategy | All the objectives have the same weight: 0.5. | It is the default strategy of the system and the control group of the experiment. |
| 2 | Clear Fitness Function | The 5 selected objectives have weight 1, the rest 0.1. | 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. |
| 3 | Slowly Finding a Fitness Function | 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. | 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. |
| 4 | Realistic | 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. | This captures the most realistic interaction, while preserving equal overall weights and allowing a comparison with the other strategies. |

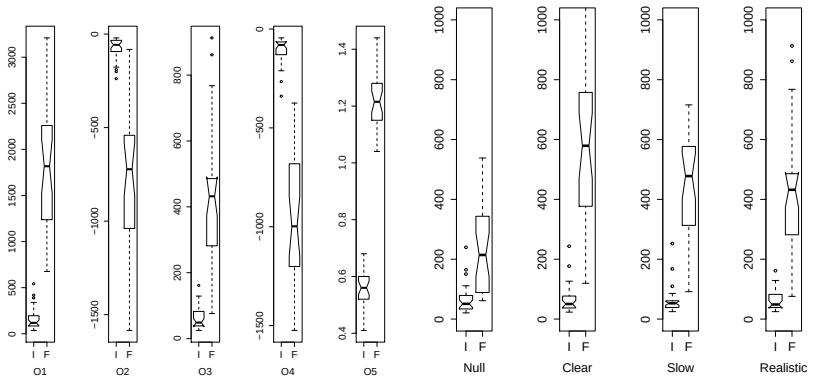
Table 3: An overview of the Interaction Strategies used for the experiment

| Step | O1 | O2 | O3 | O4 | O5 |
|---|---|---|---|---|---|
| 1 | 0.1 | 0.1 | 0.1 | 0.1 | 0.1 |
| 2 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| 3 | 0.2 | 0.2 | 0.2 | 0.2 | 0.2 |
| 4 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| 5 | 0.4 | 0.4 | 0.4 | 0.4 | 0.4 |
| 6 | 0.5 | 0.5 | 0.5 | 0.5 | 0.5 |
| 7 | 0.6 | 0.6 | 0.6 | 0.6 | 0.6 |
| 8 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 |
| 9 | 0.8 | 0.8 | 0.8 | 0.8 | 0.8 |
| 10 | 1 | 1 | 1 | 1 | 1 |
| 11 | 1 | 1 | 1 | 1 | 1 |
| $\sum$ | 6 | 6 | 6 | 6 | 6 |

(a) Strategy S3.

| Step | O1 | O2 | O3 | O4 | O5 |
|---|---|---|---|---|---|
| 1 | 0.5 | 0.5 | 0.3 | 1 | 0.6 |
| 2 | 0.4 | 0.3 | 0.2 | 1 | 0.8 |
| 3 | 0.3 | 0.1 | 0.5 | 0.8 | 1 |
| 4 | 0.2 | 0.3 | 0.8 | 0.5 | 1 |
| 5 | 0.3 | 0.5 | 1 | 0.2 | 0.6 |
| 6 | 0.4 | 0.7 | 1 | 0.2 | 0.3 |
| 7 | 0.5 | 1 | 0.8 | 0.2 | 0.1 |
| 8 | 0.6 | 1 | 0.5 | 0.3 | 0.5 |
| 9 | 0.8 | 0.7 | 0.3 | 0.5 | 0.7 |
| 10 | 1 | 0.5 | 0.2 | 0.8 | 0.3 |
| 11 | 1 | 0.4 | 0.4 | 0.5 | 0.1 |
| $\sum$ | 6 | 6 | 6 | 6 | 6 |

(b) Strategy S4.




(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 8: A detailed view of some of the experimental results.

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 8a reflect these goals.

An interesting comparison is that in Figure 8b, showing the 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.

## 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 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 8a). 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.

## 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.

## 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.

## 10. Acknowledgements

This research has been supported by funding from The Knowledge Foundation (KKS) in the project Next Generation Software Engineering (NGSE), project no. 2010/0124. It was also partly funded by the KKS project no. 20130085 Testing of Critical System Characteristics (TOCSYC).

## References

[1] P. McMinn, Search-based software testing: Past, present and future, Fourth International Conference on Software Testing, Verification and Validation Workshops (2011) 153–163.

[2] W. Afzal, R. Torkar, R. Feldt, A systematic review of search-based testing for non-functional system properties, Information and Software Technology 51 (6) (2009) 957–976. doi:10.1016/j.infsof.2008.12.005.
URL http://dx.doi.org/10.1016/j.infsof.2008.12.005

[3] R. Feldt, Genetic programming as an explorative tool in early software development phases, in: Proceedings of the 1st International Workshop on Soft Computing Applied to Software Engineering (SCASE '99), Limerick University Press, University of Limerick, Ireland, 1999, pp. 11–20.

[4] H. Takagi, Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation, Proceedings of the IEEE 89 (9) (2001) 1275 –1296. doi:10.1109/5.949485.

[5] R. Feldt, An interactive software development workbench based on biomimetic algorithms, Tech. Rep. Tech. report 02-16, Dept. of Computer Engineering, Chalmers University of Technology, Gothenburg (2002).
URL http://www.cse.chalmers.se/~feldt/publications/feldt_2002_wise_tech_report.html

[6] C. L. Simons, I. C. Parmee, R. Gwynllyw, Interactive, evolutionary search in upstream object-oriented class design, IEEE Transactions on Software Engineering 36 (6) (2010) 798–816.

[7] B. Bush, H. Sayama, Hyperinteractive evolutionary computation, Evolutionary Computation, IEEE Transactions on 15 (3) (2011) 424–433. doi:10.1109/TEVC.2010.2096539.

[8] M. Harman, B. F. Jones, Search based software engineering, Information and Software Technology (43) (2001) 833–839.

[9] S. Xanthakis, C. Ellis, C. Skourlas, A. L. Gall, S. Katsikas, K. Karapoulios, Application of genetic algorithms to software testing, in: Proceedings of the 5th International Conference on Software Engineering and Applications, Toulouse, France, 1992, pp. 625–636.

[10] M. Harman, S. A. Mansouri, Y. Zhang, Search based software engineering: A comprehensive analysis and review of trends techniques and applications, Tech. Rep. TR-09-03 (April 2009).

[11] A. Arcuri, X. Yao, Search based software testing of object-oriented containers, Information Sciences 178 (15) (2008) 3075 – 3095. doi:10.1016/j.ins.2007.11.024.

[12] S. Mairhofer, R. Feldt, R. Torkar, Search-based software testing and test data generation for a dynamic programming language, in: Proceedings of the 13th annual conference on Genetic and evolutionary computation, GECCO '11, ACM, New York, NY, USA, 2011, pp. 1859–1866. doi:10.1145/2001576.2001826.
URL http://doi.acm.org/10.1145/2001576.2001826

[13] I. C. Parmee, D. Cvetkovic, A. H. Watson, C. R. Bonham, Multiobjective satisfaction within an interactive evolutionary design environment, Evolutionary Computation 8 (2) (2000) 197–222.

[14] P. Walsh, P. Gade, Terrain generation using an interactive genetic algorithm, in: Evolutionary Computation (CEC), 2010 IEEE Congress on, IEEE, 2010, pp. 1–7.

[15] N. Hayashida, H. Takagi, Visualized iec: Interactive evolutionary computation with multidimensional data visualization, in: Industrial Electronics Society, 2000. IECON 2000. 26th Annual Conference of the IEEE, Vol. 4, IEEE, 2000, pp. 2738–2743.

[16] G. Bavota, F. Carnevale, A. De Lucia, M. Di Penta, R. Oliveto, Putting the developer in-the-loop: an interactive ga for software re-modularization, in: Proceedings of the 4th international conference on Search Based Software Engineering, SSBSE'12, Springer-Verlag, Berlin, Heidelberg, 2012, pp. 75–89.

[17] R. Kamalian, E. Yeh, Y. Zhang, A. Agogino, H. Takagi, Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems, in: Fuzzy Systems, 2006 IEEE International Conference on, 2006, pp. 678 –684. doi:10.1109/FUZZY.2006.1681784.

[18] P. Tonella, A. Susi, F. Palma, Using interactive ga for requirements prioritization, in: Proceedings of the 2nd International Symposium on Search Based Software Engineering, SSBSE '10, IEEE Computer Society, Washington, DC, USA, 2010, pp. 57–66.

[19] C. Chou, S. Kimbrough, J. Sullivan-Fedock, C. J. Woodard, F. H. Murphy, Using interactive evolutionary computation (IEC) with validated surrogate fitness functions for redistricting, in: Proceedings of the fourteenth international conference on Genetic and evolutionary computation conference, GECCO '12, ACM, New York, NY, USA, 2012, pp. 1071–1078. doi:10.1145/2330163.2330312.
URL http://doi.acm.org/10.1145/2330163.2330312

[20] X. Sun, D. Gong, Y. Jin, S. Chen, A new surrogate-assisted interactive genetic algorithm with weighted semisupervised learning, Cybernetics, IEEE Transactions on 43 (2) (2013) 685–698. doi:10.1109/TSMCB.2012.2214382.

[21] A. Liapis, G. Yannakakis, J. Togelius, Limitations of choice-based interactive evolution for game level design, in: Proceedings of AIIDE Workshop on Human Computation in Digital Entertainment, 2012.

[22] G. Avigad, A. Moshaiov, Interactive evolutionary multiobjective search and optimization of set-based concepts, Systems, Man, and Cybernetics, Part B: Cybernetics, IEEE Transactions on 39 (4) (2009) 1013–1027. doi:10.1109/TSMCB.2008.2011565.

[23] K. Deb, A. Sinha, P. Korhonen, J. Wallenius, An interactive evolutionary multiobjective optimization method based on progressively approximated value functions, Evolutionary Computation, IEEE Transactions on 14 (5) (2010) 723–739. doi:10.1109/TEVC.2010.2064323.

[24] C. L. Simons, I. C. Parmee, Elegant object-oriented software design via interactive, evolutionary computation, Systems, Man, and Cybernetics, Part C: Applications and Reviews, IEEE Transactions on 42 (6) (2012) 1797–1805.

[25] B. Marculescu, R. Feldt, R. Torkar, A concept for an interactive search-based software testing system, in: G. Fraser, J. Teixeira de Souza (Eds.), Search Based Software Engineering, Vol. 7515 of Lecture Notes in Computer Science, Springer Berlin Heidelberg, 2012, pp. 273–278. doi:10.1007/978-3-642-33119-0_21.
URL http://dx.doi.org/10.1007/978-3-642-33119-0_21

[26] A. Collins, D. Joseph, K. Bielaczyc, Design research: Theoretical and methodological issues, The Journal of the learning sciences 13 (1) (2004) 15–42.

[27] P. Bentley, J. Wakefield, Finding acceptable solutions in the pareto-optimal range using multiobjective genetic algorithms, in: P. Chawdhry, R. Roy, R. Pant (Eds.), Soft Computing in Engineering Design and Manufacturing, Springer London, 1998, pp. 231–240. doi:10.1007/978-1-4471-0427-8_25.

[28] R. Storn, K. Price, Differential evolution a simple and efficient heuristic for global optimization over continuous spaces, J. of Global Optimization 11 (4) (1997) 341–359. doi:10.1023/A:1008202821328.
URL http://dx.doi.org/10.1023/A:1008202821328

[29] M. Črepinšek, S.-H. Liu, M. Mernik, Replication and comparison of computational experiments in applied evolutionary computing: Common pitfalls and guidelines to avoid them, Applied Soft Computing 19 (0) (2014) 161 – 170. doi:http://dx.doi.org/10.1016/j.asoc.2014.02.009.
URL http://www.sciencedirect.com/science/article/pii/S1568494614000787