Applied Soft Computing 29 (2015) 26–39

Contents lists available at ScienceDirect

Applied Soft Computing

journal homepage: www.elsevier.com/locate/asoc

# 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

ARTICLE INFO

*Article history:*
Received 7 February 2014
Received in revised form 4 December 2014
Accepted 16 December 2014
Available online 25 December 2014

*Keywords:*
Search-based software testing
Interactive search-based software engineering
User centered
Embedded software
Industrial experience

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 were 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.

## 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]

* Corresponding author. Tel.: +46 734223506.
*E-mail addresses:* marculescu.bogdan@gmail.com, bogdan.marculescu@bth.se (B. Marculescu).

[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.

http://dx.doi.org/10.1016/j.asoc.2014.12.025
1568-4946/© 2014 Elsevier B.V. All rights reserved.

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 [3–7], 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 used by domain specialists to develop test cases without requiring extensive training in its use.
- 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. [3,9,10]. 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 toward 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