# A Concept for an Interactive Search-Based Software Testing System

Bogdan Marculescu, Robert Feldt, and Richard Torkar

Blekinge Institute of Technology,
School of Computing
Karlskrona, Sweden
`bogdan.marculescu@bth.se`

**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 assembled, and sometimes even developed, 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 a 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 being used to determine the fitness of potential solutions.

**Keywords:** search-based software testing, interactive search-based software engineering, user centered

## 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 no 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 [1, 2]. 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 allows 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   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 [1, 2], from object-oriented containers [3] to dynamic programming languages [4]. SBST is a part of the wider area of Search-Based Software Engineering (SBSE), a term coined by Harman and Jones in 2001 [5], but a concept used earlier, see e.g. [6–8]. Since search-based approaches have been applied across the software development life-cycle [9], 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.

# 3    Interactive Search-Based Testing System

## 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.

## 3.2    Overview and Components

The figure 1 shows the structure of a complex Interactive Search-Based Software Testing system developed using the proposed methodology.
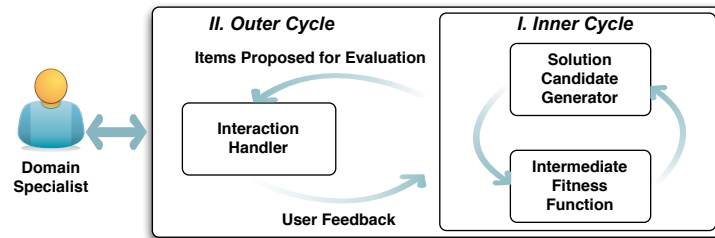


**Fig. 1.** Overview of an ISBSE system.

*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 is meant to fully benefit from exploratory capabilities of search-based systems [10], while still allowing 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 is achieved through the populations of candidates, the evaluations made by the domain expert and the feedback that guide the Intermediate Fitness Function. The generation and selection of the population, as well as the internal workings of the inner cycle are hidden from the domain expert.

## 4   Validation and Discussion

Validation efforts are focused on the development of a proof of concept system. This system is being 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.

## 5   Conlusions

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.

## References

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

2. Afzal, W., Torkar, R., Feldt, R.: A systematic review of search-based testing for non-functional system properties. Information and Software Technology (2009)
3. Arcuri, A., Yao, X.: Search based software testing of object-oriented containers. Information Sciences **178**(15) (2008) 3075 – 3095
4. Mairhofer, S., Feldt, R., Torkar, R.: 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, New York, NY, USA, ACM (2011) 1859–1866
5. Harman, M., Jones, B.F.: Search based software engineering. Information and Software Technology (43) (2001) 833–839
6. Xanthakis, S., Ellis, C., Skourlas, C., Gall, A.L., Katsikas, S., Karapoulios, K.: Application of genetic algorithms to software testing. In: Proceedings of the 5th International Conference on Software Engineering and Applications, Toulouse, France (7-11 December 1992) 625–636
7. Feldt, R.: Generating multiple diverse software versions with genetic programming - an experimental study. IEE Proceedings - Software **145**(6) (December 1998) 228–236
8. Harman, M., Mansouri, S.A., Zhang, Y.: Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03 (April 2009)
9. Harman, M.: The current state and future of search based software engineering. Future of Software Engineering (FOSE'07) (2007)
10. Feldt, R.: 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), University of Limerick, Ireland, Limerick University Press (12-14 April 1999) 11–20
11. Simons, C.L., Parmee, I.C., Gwynllyw, R.: Interactive, evolutionary search in upstream object-oriented class design. IEEE Transactions on Software Engineering **36**(6) (November/December 2010) 798–816
12. Simons, C., Parmee, I.: User-centered, evolutionary search in conceptual software design. In: Evolutionary Computation, 2008. CEC 2008. (IEEE World Congress on Computational Intelligence). IEEE Congress on. (june 2008) 869 –876
13. Farrington, J.: Seven plus or minus two. Performance Improvement Quarterly **24(4)** (2011) 113116
14. Tarnow, E.: There is no capacity limited buffer in the murdock (1962) free recall data. Cognitive Neurodynamics **4** (2010) 395–397 10.1007/s11571-010-9108-y.
15. Metzger, U., Parasuraman, R.: Automation in future air traffic management: Effects of decision aid reliability on controller performance and mental workload. Human Factors: The Journal of the Human Factors and Ergonomics Society **47**(1) (Spring 2005) 35–49
16. Maguire, M.: Methods to support human-centred design. International Journal of Human-Computer Studies **55**(4) (2001) 587 – 634
17. Takagi, H.: Interactive evolutionary computation: fusion of the capabilities of ec optimization and human evaluation. Proceedings of the IEEE **89**(9) (sep 2001) 1275 –1296
18. Kamalian, R., Yeh, E., Zhang, Y., Agogino, A., Takagi, H.: Reducing human fatigue in interactive evolutionary computation through fuzzy systems and machine learning systems. In: Fuzzy Systems, 2006 IEEE International Conference on. (0-0 2006) 678 –684
19. Feldt, R.: An interactive software development workbench based on biomimetic algorithms. Technical Report 02-16, Gothenburg, Sweden (November 2002)