The Relationship between Search Based Software Engineering and Predictive Modeling

Mark Harman
University College London,
Department of Computer Science, CREST Centre,
Malet Place, London, WC1E 6BT, UK.

ABSTRACT

Search Based Software Engineering (SBSE) is an approach to software engineering in which search based optimization algorithms are used to identify optimal or near optimal solutions and to yield insight. SBSE techniques can cater for multiple, possibly competing objectives and/ or constraints and applications where the potential solution space is large and complex. This paper will provide a brief overview of SBSE, explaining some of the ways in which it has already been applied to construction of predictive models. There is a mutually beneficial relationship between predictive models and SBSE. The paper sets out eleven open problem areas for Search Based Predictive Modeling and describes how predictive models also have role to play in improving SBSE.

1. INTRODUCTION

Harman and Clark argued that 'metrics are fitness functions too' [54]. This paper extends this analogy to consider the case for 'predictive models are fitness functions too'. It turns out that the relationship between fitness functions and predictive models is much richer than this slightly facile aphorism might suggest. There are many connections and relationships between Search Based Software Engineering and Predictive Modeling. This paper seeks to uncover some of these and describe ways in which they might be exploited, both by the SBSE community and by the Predictive Modeling community.

Search based Software Engineering (SBSE) seeks to reformulate Software Engineering problems as search problems. Rather than constructing test cases, project schedules, requirements sets, designs, architectures and other software engineering artifacts, SBSE simply searches for them. The search space is the space of all possible candidate solutions. This is typically enormous, making it impossible to enumerate all solutions. However, guided by a fitness function that distinguishes good solutions from bad ones, we can automate the search for good solutions in this search space.

SBSE has witnessed a dramatic increase in interest and activity in the last few years, resulting is several detailed surveys of techniques, applications and results [2, 9, 50, 57, 60, 100]. It has been shown that SBSE solutions to Software Engineering problems are competitive with those produced by humans [29, 115] and that the overall SBSE approach has tremendous scalability potential [13, 49, 85, 94]. Section 2 provides a very brief overview of SBSE, while Section 3 present an overview of previous work on SBSE applied to predictive modeling.

SBSE is attractive because of the way it allows software engineers to balance conflicting and competing constraints in search spaces characterized by noisy, incomplete and only partly accurate data [50, 53]. This characterization of the problem space may sound surprisingly familiar to a reader from the Predictive Modeling community. Predictive modeling also has to contend with noisy, incomplete and only partly accurate data. As a result, there is a great deal of synergy to be expected between the two communities. This paper seeks to explore this synergy. It will also argue that the way in which SBSE can cater for multiple competing and conflicting objectives may also find, as yet largely unexplored, applications in predictive modeling work.

The 2009 PROMISE conference call for papers raised four questions for which answers were sought from the papers submitted. Clearly, in the year that has passed since PROMISE 2009, progress will have been made. However, the general character of the four questions means that they are unlikely to be completely solved in the near future. Since these questions are likely to remain open and important for the predictive modeling research community they serve as a convenient set of questions for this paper to seek to answer through the perspective of SBSE. The questions, and the sections that address them, are as follows:

1. How much do we need to know about software engineering in order to build effective models? (addressed in Section 4)
2. How to adapt models to new data? (addressed in Section 5)
3. How to streamline the data or the process? (addressed in Section 6)
4. How can predictive modeling gain greater acceptance by the SE community? (addressed in Section 7)

Of course, it would be one-sided and simplistic to claim that the predictive modeling community had so much to gain from considering SBSE, without also considering the reciprocal benefit to the SBSE community that may come from results and techniques associated with predictive modeling. Indeed, there are several ways in which predictive modeling may also benefit the SBSE community. These are briefly set out in Section 8.

The paper is written as an accompaniment to the author's keynote presentation at PROMISE 2010: The 6^{th} International Conference on Predictive Models in Software Engineering. The paper, like the keynote, is constructed to raise questions and possibilities, rather

PROMISE2010, Sep 12-13, 2010. Timisoara, Romania

than to answer them. The paper presents 11 broad areas of open problems in SBSE for predictive modeling, explaining how techniques emerging from the SBSE community may find potentially innovative applications in Predictive Modeling. These open problem areas (and the sections that describe them) are as follows:

1. Multi objective selection of variables for a predictive model (Section 4)
2. Sensitivity analysis for predictive models (Section 4)
3. Risk reduction as an optimization objective for predictive models (Section 4)
4. Exploiting Bayesian models of Evolutionary Optimization (Section 5)
5. Balancing functional and non–functional properties of predictive models (Section 6)
6. Exploiting the connection between fitness function smoothing and model interpolation (Section 6)
7. Validating predictive models through optimization (Section 7)
8. Human–in–the–loop predictive Modeling with Interactive Evolutionary Optimization (Section 7)
9. Identifying the building blocks of effective predictive models (Section 7)
10. Predictive models as fitness functions in SBSE (Section 8)
11. Predictive models of SBSE effort and landscape properties (Section 8)

2. SEARCH BASED SOFTWARE ENGINEERING

Software Engineering, like other engineering disciplines, is all about optimization. We seek to build systems that are better, faster, cheaper, more reliable, flexible, scalable, responsive, adaptive, maintainable, testable; the list of objectives for the software engineer is a long and diverse one, reflecting the breadth and diversity of applications to which software is put. The space of possible choices is enormous and the objectives many and varied.

In such situations, software engineers, like their peers in other engineering disciplines [117] have turned to optimization techniques in general and to search based optimization in particular. This approach to software engineering has become known as Search Based Software Engineering [50, 57]. Software Engineering is not merely yet another engineering discipline for which Search Based Optimization is *suitable*; its virtual nature makes software the *ideal* 'killer application' for search based optimization [53].

SBSE has proved to be very widely applicable in Software Engineering, with applications including testing [16, 18, 23, 56, 81, 90, 113], design, [26, 55, 93, 104, 110], requirements, [14, 39, 38, 66], project management [3, 10, 11] refactoring [61, 95] and, as covered in more detail in the next section, predictive modeling. To these Software Engineering problems a wide range of different optimisation and search techniques have been applied such as local search [73, 85, 92], simulated annealing, [21, 15, 92], Ant Colony Optimization [7, 91], Tabu search [31], and most widely of all (according to a recent review [60]), genetic algorithms and genetic programming [17, 46, 55, 113]. Exact and greedy algorithms, though not strictly 'search' based are also often applied to similar problems, for example to requirements [122] and testing [119]. As such, these algorithms are also regarded by many researchers as falling within the remit of the SBSE agenda.

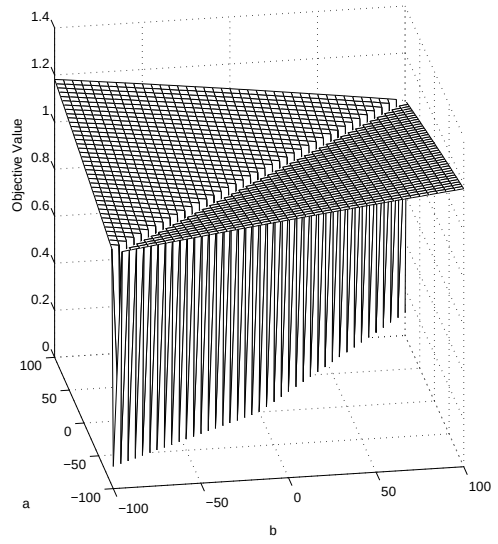
The reader who is unfamiliar with these algorithms, can find overviews and more detailed treatments in the literature on SBSE [60, 50, 100]. For the purpose of this paper, all that is required is to note that SBSE applies to problems in which there are numerous candidate solutions and where there is a fitness function that can guide the search process to locate reasonably good solutions. The algorithms in the literature merely automate the search (using the guide provided by the fitness function in different ways). For example, genetic algorithms, inspired by natural selection, evolve a population of candidate solutions according to the fitness function, while hill climbing algorithms simply take a random solution as a starting point and apply mutations to iteratively seek out near neighbours with improved fitness. As has been noted in the literature on SBSE, the reader can interpret the words 'fitness function' as being very closely related to 'metric' in the Software Engineering sense of the word [54].

It is important not to be confused by terminology: In Search Based Software Engineering, the term 'search' is used to refer to 'search' as in 'search-based optimization' not 'search' in the sense of a 'web search' or a 'library search'. It should also be understood that search seldom yields a globally optimal result. Typically a 'near optimal' solution is sought in a search space of candidate solutions, guided by a fitness function that distinguishes between better and worse solutions. This near optimal solution, merely needs to be 'good enough' to be useful. SBSE is very much an 'engineering approach' and so SBSE researchers often formulate problems in terms of finding solutions that are good enough, better than previously obtained or within acceptable tolerances.

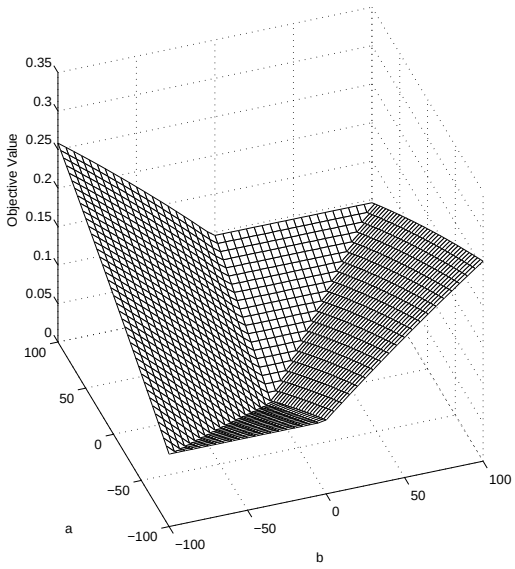
A detailed survey of all work on SBSE up to December 31st 2008 can be found in the work of Harman et al. [60]. There are also more in-depth surveys on particular aspects of SBSE, for example testing [9, 86] design level SBSE [100] and SBSE for non–functional properties [2].

It is a useful conceptual device to imagine all possible solutions (including the very worst and best) as lying on a horizontal plain in which similar candidates are proximate. We can think of the fitness of each individual in the plain as being denoted by a vertical high in the vertical dimension. In this way we conceptualize a 'fitness landscape', in which any globally optimal solutions will correspond to the highest peaks and in which properties of the landscape may relate to the difficulty of finding solutions using a chosen search technique. If the problem is one of maximizing the value of fitness then we seek to find solutions that denote the tops of hills in the landscape. If the problem is to minimize the value of the fitness function, then we seek the deepest troughs. Figure 1 presents an example of two such fitness landscapes, in which the goal is to minimize the value of fitness. The visualization of fitness landscapes provides insights into the nature of the search problem. In Figure 1 the upper landscape is far more challenging than the lower landscape. The lower landscape is produced by transforming the upper landscape, thereby making the search more efficient and effective. Sections 6 and 8 consider the ways in which such transformations may be useful to Predictive Modeling, and how predictive modeling may help SBSE researchers to better understand their search landscapes.

Like other engineers, software engineers find themselves unable to achieve all possible objectives and therefore some balance between them is typically sought. This motivates a multi objective approach to optimization which is increasingly prevalent in SBSE work. Search based optimization is naturally suited to the balancing of competing and conflicting multiple objectives and so these



Untransformed program



Transformed version

Figure 1: Transforming the Fitness Landscape by Transforming the Fitness Function (example from previous work [88, 89]). In this example, we seek the global minimum in the search space. The untransformed landscape is exceptionally difficult: The global minimum is found when the value of a and b are both -1. However, in all other cases where the value of a and b are equal one of the local minima is reached. The fitness function is transformed in this case by transforming the program from which fitness is computed. As a result, the local minima disappear from the landscape and all points on the landscape guide the search towards the global minimum. A problem can be transformed so that search will be dramatically more likely to locate a global optima, without necessarily knowing the values denoted by this optimum. As such, this transformation is an effective approach to migrating a hard search problem into an easier search problem from which optimal solutions can be found. Such approaches may also find applications in smoothing and optimizing predictive models.

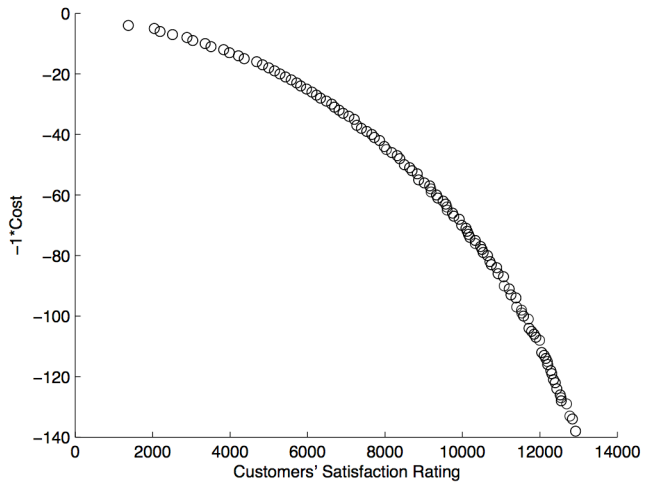


Figure 2: A cost–benefit Pareto front from Search Based Requirements Optimization [122]. Moving up the vertical axis denotes reduced cost; moving rightward on the horizontal axis denotes increased customer satisfaction. Each circle is one valid choice of requirements that minimizes cost and maximizes value. For the set of requirements denoted by each circle, the search can find no solution with a lower cost and greater value. We can use this to find a good trade off between cost and value. This approach can be adapted to Search based Predictive Modeling in order to balance competing and conflicting predictive modeling objectives. For example, it could be used to minimize risk while maximizing predictive quality. Multi objective predictive modeling possibilities are discussed in more detail in Sections 4 and 6.

multi objective approaches have been adopted across the spectrum in software quality [72], testing [78, 30, 36, 59, 120], requirements [41, 102, 123], design [22, 98], and management [8, 12]. Sections 4 and 6 claim that multi objective SBSE also has an important role to play in optimizing predictive models for several competing and conflicting objectives.

Figure 2 gives an example of a multi objective solution Pareto front from the domain of Requirements Optimization. The solution is a Pareto front of solutions each of which denotes a non–dominated solution. A solution is Pareto optimal if there is no other solution which better solves one of the two objectives without also giving a worse solution for the other objective. In this way, the Pareto front contains a set of choices, each of which balance in an optimal manner, the trade off between the functional and non–functional properties.

Like all metaheuristic search, one cannot be sure that one has the true globally optimal Pareto front and so we seek to produce as good an approximation as possible. Nevertheless, the Pareto front can yield insights into the tradeoffs present.

3. PREVIOUS WORK ON SBSE FOR PREDICTIVE MODELING

SBSE techniques have a natural application in Predictive Modeling. In order to apply an SBSE approach to prediction, we need to capture the objectives of the predictive modeling system as a set of fitness functions which the SBSE algorithm can optimize. One

natural way in which this can be done is to use the degree of fit of an equation to a set of data points as a fitness function.

The equation is sought from among a very large set of candidate equations, using a Genetic Programming (GP) approach. In this approach, the equation is not constrained to use only certain combinations of expressions and does not suffer from the limitations of a standard linear regression fit, in which a comparatively limited set of possibilities are usually available.

As with predictive modeling in general, software cost estimation is one of the areas in which this search based predictive modeling approach has been most widely studied. Software project cost estimation is known to be a very demanding task [106] with inherent uncertainties and a lack of reliable estimates. Software cost estimation is further hampered by poor quality, missing and incomplete data sets. All of these problems are very well addressed by SBSE. The SBSE approach is able to handle missing and incomplete data and can seek a best fit equation that may be piecewise; revealing very different trends (and requiring consequently different equations) for different parts of the problem space.

The first author to use a search based approach to project cost estimation was Dolado [32, 33, 34]. He used a GP approach to evolve functions that fitted the observed data for project effort (measured in function points).

In Dolado's work, the population consists of a set of equations. The operators that can be included in these GP-evolved equations are very inclusive. Dolado includes arithmetic operators and other potentially useful mathematical functions such as power, square roots and log functions. Dolado's fitness function is the *mean squared error*,

$$mse = \frac{1}{n-2}\sum_{i=1}^{n}(y_i - \hat{y}_i)^2$$

Several other authors have also used GP to fit equations that predict quality, cost and effort [19, 20, 24, 37, 69, 70, 71, 80, 82, 83, 84, 108]. Other authors have compared GP to more 'classical' techniques such as linear regression [105] and have combined search based approaches with approaches that use Artificial Neural Networks [109] and Bayesian classifiers [21].

SBSE has also been used to build predictive models for defect prediction [67], performance prediction [75] and for the classification [112] and selection [73] of the metrics to be used in prediction. However, SBSE has been much less applied to predicting locations and numbers of bugs [1], which is surprising considering how important this is within the predictive modeling community [96].

4. CAN SBSE TELL US HOW MUCH DO WE NEED TO KNOW IN ORDER TO BUILD EFFECTIVE MODELS?

Building an effective model that can produce reasonable predictions of a dependent variable set V requires at least two important ingredients:

1. There has to be a set of variables A, the values of which determine or at least influence the values of V.
2. There has to be sufficient information about values of A in order to support the inference of a reasonable approximation to the relationship that exists between A and V.

This may seem very obvious, but it is often overlooked or inadequately accounted for. Given a candidate set of 'influencing factors' (which, one hopes to be a superset of A) it is not unreasonable to ask

> "Which of the candidate variables' values can affect the values of the variables we seek to predict?"

This problem can be addressed in a variety of ways, one of which involves an SBSE approach. Kirsopp et al. [73] showed how a simple local search could be used to find those sets of project features that are good predictors of software project effort and cost. They demonstrated how their approach could assist the performance of Case Based Reasoning systems by identifying, from a set of 43 possible candidate project variables, those which are good predictors.

This approach is attractive because it is generic. For a candidate set, A', of prediction variables to be included in the model, the search must allocate a fitness to A'. Kirsopp et al. simply 'jack knife' the set of data to treat each project as a possible candidate for prediction. They discard information about the chosen project and seek to predict the discarded values using A'. This gives a fitness based on prediction quality. The experiment can be repeated for each project in the set to give an overall estimate of the predictive quality of the set A'. The aggregate predictive quality denotes the fitness of the choice of variables A'. In general, the closer A' is to A the better should be its fitness.

In a set of 43 variables there are 2^{43} possible subsets so the prediction variable space is too large to enumerate. However, using a simple local search Kirsopp et al. were able to identify good sets of prediction variables. They were also able to use the simplicity of their hill climbing search technique to good advantage in determining properties of the search space. They examined the fitness landscape to reveal that the search problem had large basins of attraction and clustered peaks, making hill climbing a reasonable search technique to apply.

Their approach was evaluated with respect to Case Based Reasoning but this was not an inherent aspect of their approach. It would be possible to use a similar approach to locate the set of 'useful prediction variables' for any predictive modeling approach. Furthermore, there are other factors that could be taken into account. Kirsopp et al. published their results in 2002. Since that time, there has been an increasing interest in multi objective optimization for software engineering (as mentioned in the introduction to this paper).

A multi objective approach to the problem of identifying a good prediction set could take into account other factors as well as prediction quality. There may be several factors that are important in a good model, and there may be tensions between these factors because the achievement of one may be won only at the expense of another (a very typical problem for multi objective optimization). Such additional objectives for predictive modeling that a multi objective search based approach might seek to optimize include:

1. **Predictive Quality**. Clearly we shall want our predictions to be reasonably accurate in order to be useful. Prediction quality may not be a single simple objective. There are many candidates for measuring prediction quality. Typically [24, 33] MMRE has been used. However, this assumes that we simply wish to minimize the likely error. This may not always be the case. Also, MMRE is only one of many possible metrics and there has been evidence to suggest that it may not always select the best model [43].

   It has also been argued that information theoretic measures have an important role to play in SBSE as fitness functions [50]. Such measures can capture information content and information flow and make ideal fitness function candidates because the solutions to be optimized in SBSE are typically composed of information [53]. For measuring fitness of pre-

dictive models we may look to techniques founded on information theory such as the Akaike Information Criterion [4] and Schwartz' Bayesian variant [103].

A prediction model that is often very accurate, but occasionally exceptionally inaccurate may not be desirable though it may have an apparently attractive MMRE. The decision maker may seek a prediction system that reduces the risk of an inaccurate prediction, even at the expense of overall predictive quality. There has been previous work on risk reduction for project planning using SBSE [12], but there has been no work on risk as an objective to be minimized in search based predictive modeling; the problem of incorporating risk into search based models therefore remains open.

2. **Cost**. There may be costs associated with applying a predictive model to obtain a prediction, introducing a cost–benefit aspect to the optimization problem: we seek the best prediction we can for a given cost. Cost–benefit analysis is useful where there is a cost of collecting data for an attribute of interest. For example, it may be difficult or demanding to obtain some data, but relatively easy to obtain other data. There may also be a monetary cost associated with obtaining information, for example when such data is obtained from a third party. In such situations we have a cost–benefit trade off for which a multi objective search is ideally suited. However, this has not previously been explored in the SBSE literature.

3. **Privacy**. Certain information may have degrees of privacy and/or access control that may limit the applicability of the model. It may be preferable to favour, wherever possible, less private data in constructing or applying a predictive model. Once again, a multi objective search based optimization could handle this situation in a very natural manner, provided the degree of privacy could be measured and, thereby, incorporated into a fitness function.

4. **Readability** Predictive models may be used to give insight to the developer, in which case the readability of the model may be important. Tim Menzies kindly pointed out that readability and concision may be possible objectives in response to an earlier draft of this paper, also suggesting how they might be measured [45].

5. **Coverage and Weighting**. We may have several dependent variables in V, for which a prediction is sought. It could be that we can find sets of predictors that predict one of the outcomes in V better than another. In such cases we may seek a solution that is a good generalist, balancing all of the dependent variables equally and seeking to predict all equally well. Alternatively, some predicted variables may be more important and we may therefore allocate a higher weight to these.

   Both these situations can be covered in a multi objective search, using standard search based optimization techniques. For coverage of a set of predictors, with equal weighting or where we are unsure about the relative weights of each, we can adopt a Pareto optimal approach, which treats all objectives as equal. On the other hand, where we know the relative weighting to be applied to the elements of V, we can simply use a single objective approach in which the overall fitness is a weighted sum of the predictive quality of each element in V.

For complex, multiple objective problems such as these, where the constraints may be tight and the objectives may be conflicting and competing, multi objective SBSE has proved to be exceptionally effective. It remains an open problem to apply multi objective SBSE to the problems of predictive modeling. In many cases, a Pareto optimal approach will be appropriate because the objectives will be competing and conflicting and there will be no way to define suitable weights to balance them. Using a Pareto optimal approach, we can obtain a Pareto front similar to that depicted in Figure 2 that balances, for example, privacy against predictive quality or cost against predictive quality. Such graphs are easier to visualize in two dimensions, but it is also possible to optimize for more than two objectives.

In addition to identifying which sources of data and which variables are important for predictive modeling, there is also a related problem of determining which of the predictor variables in A will have the greatest impact on the predictions that the model produces. For these highly 'sensitive' variables, we shall need to be extra careful. This is particularly important in Software Engineering predictive models, because it is widely known that estimates in Software Engineering are often very inaccurate [99, 107, 111].

If we can identify those sensitive elements of A that may have a disproportionately large impact on the solution space, then we can pay particular attention to ensuring that the estimates of these elements is as accurate as possible. This may save time and may increase uptake as a result. It may be cost effective to focus on the top 10% of input variables that require careful prediction rather than paying equal attention to all input variables. Similarly we can prioritize the input variables so that more effort is spent calibrating and checking the estimates of those for which the model is most sensitive.

There has been previous work on such sensitivity analysis for problems in Requirements Optimization [58]. Hitherto, this approach to search based sensitivity analysis remains unexplored for predictive modeling.

5. CAN SBSE HELP US TO ADAPT MODELS TO NEW DATA?

Re-estimating the values of some 'sensitive' variables will present us with new data. The new data will override untrusted existing data because they are better estimated and therefore preferable. However, new data is not always a replacement for existing poor quality data. It may be that existing data is more trusted and new data is viewed with healthy scepticism until it has been demonstrated to provide reliable prediction. In this situation, a model needs to take account of the relative lack of trust in the new data. One obvious way in which this can be achieved is to incorporate a Bayesian model into the prediction system [40].

Over the past decade, researchers in the optimization community have considered models of Evolutionary Computation that involve statistical computations concerning the distribution of candidate solutions in a population of such solutions. This trend in research on Estimation of Distributions Algorithms' (EDAs) may have relevance to the predictive modeling community. In particular, two classes of EDAs are worth particular attention: the Bayesian Evolutionary Algorithms (BEA) and the Bayesian Optimization Algorithms (BOA). Both may have some value to the Predictive Modeling community.

In a BEA [121], the evolutionary algorithm seeks to find a model of higher posterior probability at each iteration of the evolutionary computation, starting with the prior probabilities. One of the outputs of a BEA is a Bayesian network. In this way, a BEA can be considered to be one way in which a search based approach can be used to construct a Bayesian Network.

A BOA [97] uses Bayesian modeling of the population in order to predict the distribution of promising solutions. Though not essential for the approach to work, the use of Bayesian modeling allows for the incorporation of prior knowledge into the search concerning good solutions and their building blocks. It is widely observed that the inclusion of such domain knowledge can improve the efficiency of the search. The BOA provides one direct mechanism through which this domain knowledge can be introduced.

6. CAN SBSE HELP US TO STREAMLINE THE DATA OR THE PROCESS?

SBSE techniques may be able to streamline the predictive modeling process and also the data that is used as input to a model. This section explains how SBSE has been used to trade functional and non–functional properties and how this might yield insight into the trade offs between functional and non–functional properties for predictive models.

The section also briefly reviews the work on fitness function smoothing and discusses how this may also have a resonance in work on predictive modeling.

Section 4 briefly discussed the open problem of multi–objective predictive model construction using SBSE to search for models that balance many of the competing objectives that may go towards making up a good model. However, the objectives considered could all be characterized as being 'functional' objectives; they concern the predictive quality, cost and characteristics of the predictions made by the model. We may optimize for non–functional properties of the predictive model and may also seek a balance between functional and non–functional aspects of the model.

There has been a long history of work on the optimization of non–functional properties, such as temporal aspects of software testing [6, 114] and stress testing [5, 23, 44]. Afzal *et al.* [2] provide a survey of non–functional search based software testing. Recent work on SBSE for non–functional properties [68, 116], has also shown how functional and non–functional properties can be jointly optimized in solutions that seek to balance their competing needs.

White et al. [116] show how functional and non–functional properties can be traded for one another. They use GP to evolve a pseudo-random number generator with low power consumption. They do this by treating the functional property (randomness) and the non–functional property (power consumption) as two objectives to be solved using a Pareto optimal multi objective search. Using the Pareto front, White et al. were able to explore the operators that had most impact on the front, thereby gaining insight into the relationship between the operations used to define a pseudo random number generator and their impact on functional and non–functional properties.

In using SBSE as an aid to predictive modeling we re-formulate the question 'how should we construct a model' to 'what are the criteria to guide a search for the best model among the many candidates available'. This is a 'reformulation' of the software engineering problem of predictive modeling as a search problem in the style of Clark et al. [25]. With such a reformulation in mind, there is no reason not to include non-functional aspects of the model in the fitness function which guides the search.

In this way, we could search for a model that balances predictive quality against the execution time of the model. For some applications, the execution time of certain model aspects may be high. This could be a factor in determining whether a predictive model is practical. In such situations it would be advantageous to factor into the search the temporal aspects of the problem. Using such a multi objective approach, the decision maker could consider a Pareto front of trade offs between different models, each of which optimally balances the choice between predictive quality and time to produce the prediction.

As with the work of White et al. [116], the goal may not be necessarily to produce a solution, but to yield *insight* into the trade offs inherent in the modeling choices available. Producing a sufficiently accurate and fast predictive model may be one possible aim, but we should not overlook the value of simply understanding the balance of trade offs and the effect of modeling choices on solution characteristics. SBSE provides a very flexible and generic way to explore these trade offs.

It is not uncommon for approaches to search based optimization to estimate or approximate fitness, rather than computing it directly. Optimizers can also interpolate between fitness values when constructing a new candidate solution from two parents. This is not merely a convenience, where only approximate fitness is possible. Rather, it has been shown that smoothing the search landscape can improve the performance of the search [101]. Some of the techniques used for fitness approximation and interpolation either implicitly or explicitly accept that the fitness computed may not be entirely reliable. There would appear to be a strong connection between such realistic acceptance of likely reliability and the problem of constructing a predictive model. Perhaps some of the techniques used to smooth fitness landscapes can be adapted for use in predictive modeling.

7. CAN SBSE HELP PREDICTIVE MODELING GAIN GREATER ACCEPTANCE BY THE SE COMMUNITY?

As described in Section 6, SBSE may not be used merely to help find good quality predictive models from among the enormous space of candidates. It may also be used as a tool for gaining insight into the trade offs inherent in the choices between different model characteristics, both functional and non–functional.

We might consider a predictive model to be a program. It takes inputs in the form of data and produces output in the form of predictions. The inputs can be noisy, imprecise and only partly defined and this raises many problems for modeling as we have discussed earlier. The outputs are not measured according to correctness, but accuracy, forcing a more nuanced statistical view of correctness. Both these input and output characteristics make predictive models unlike traditional programs, but this should not obscure our recognition that predictive models are a form of program.

This observation underpins the application of Genetic Programming (GP) (as briefly reviewed in Section 3). We know from the literature on GP [79, 74] that GP is good at providing small programs that are nearly correct. Scaling GP to larger programs has proved to be a challenge, though there has been recent progress in this area [65, 78]. However, such GP scalability issues are not a problem for predictive modeling, because the programs required, although subtle, are not exceptionally long. Furthermore, traditional programming techniques cope inadequately with ill-defined, partial and messy input data, whereas, this is precisely the area where the GP approach excels.

In many ways, the GP community and the predictive modeling communities have a similar challenges to overcome with regard to acceptance within the wider Software Engineering community. The *perceived* 'credibility gap', in part, may stem from the very nature of the problems both communities seek to tackle. Messy and noisy inputs are deprecated in Software Engineering and much effort has gone into eliminating them. Both the GP and the predictive model-

ing communities are forced to embrace this 'real world messiness' and do not have the luxury of simply rejecting such inputs as 'invalid'. Similarly, outputs for which no 'correctness guarantee' can be provided sit uneasily on the shoulders of a science that grew out of mathematics and logic and whose founders eschewed any such notions [27, 42, 62].

SBSE researchers face a similar challenge to that faced by researchers and practitioners working with predictive models. SBSE is all about engineering optimization. Its motivation and approach are inherently moulded by an engineering mindset, rather than a more traditional mindset derived from mathematical logic. As such, SBSE is concerned with im*proving* not with *proving*. Perhaps the same can be said of Predictive Modeling.

Fortunately there is a growing acceptance that Software Engineers are ready to widen their understanding of correctness (and its relationship to *usefulness*). Increasingly, software engineering is maturing to the point where it is able to start to consider complex messy real world problems, in which the software exhibits engineering characteristics as well as mathematical and logical characteristics [63]. The growing interest in SBSE may be one modest example of this increasing trend [60], though it can also be seen in the wide acceptance of other useful (though not guaranteed correct) techniques, such as dynamic determination of likely invariants [35].

There are several ways in which SBSE researchers have found it possible to increase acceptability of search based solutions and these may extend to predictive modeling, because of the close similarities outlined above. Four possible approaches to gaining greater acceptance using SBSE are outlined below:

1. **Justification**. It may be thought that SBSE is a form of 'Artificial Intelligence for Software Engineering'. However, this observation can be misleading. While it is true that some search techniques have been used for AI problems, the application of search based optimization for Software Engineering is all about optimization and little about artificial intelligence. One problem with an 'AI for SE' view point is that some AI techniques lack an adequate explanation for the results they produce. This is a common criticism of Artificial Neural Net approaches to predictive modeling [64].

   SBSE is not about artificial intelligence; the only 'intelligence' in the search process is supplied, *a priori*, by the human in the form of the fitness function and problem representation and in the construction of the search algorithm. Also, by contrast with ANNs and other so-called 'black box' AI techniques, the SBSE approach is far from being 'black box'. Indeed, one of the great advantages of the SBSE approach is precisely that it is *not* a 'black box' approach. Algorithms such as hill climbing and evolutionary algorithms make available a *detailed* explanation of the results they produce in the trace of candidates considered. For example, in hill climbing, each solution found is better than the last, according to the fitness function. This means that there is a sequence of reasoning leading from one individual to the next, which explains how the algorithm considered and discarded each candidate solution on the way to its final results.

2. **Human–in–the–loop** Search based optimization is typically automated. It is often the ability to automate the search process that makes SBSE attractive [50, 53]. However, the search process does not have to be *fully* automated. It is possible to employ the human to guide fitness computation while the algorithm progresses. For example, interactive evolution computes fitness, but allows the human to make a contribution to the fitness computation, thereby incorporating human knowledge and judgement into the evolutionary search process.

   Interactive evolution provides a very effective way to allow the human a say in the construction of solutions. It may be essential to do this, perhaps because fitness cannot be wholly captured algorithmically or because there is a necessary element of aesthetics or subjective judgement involved. In previous work on SBSE the possibility of interactive evolution has been proposed for applications to program comprehension [51] and design [110]. It could also be used to increase acceptance of solutions; if an engineer has helped to define fitness, might such an engineer not be more likely to accept its optimized results?

3. **Optimization as Evaluation** Using SBSE, any metric can be treated as a fitness function [54]. Therefore, if we have a candidate prediction system, we can use it as a fitness function and optimize solutions according to its determination of high values from low. Essentially, we are using the prediction system 'in reverse'; we seek to find those sets of decision variables in A that lead to high values of one of the predicted variables in V. This could have interesting and useful applications in the validation of predictive models.

   Many SBSE techniques can be configured to produce a sequence of candidate solutions with monotonically increasing fitness. This is one of the advantages of the way a search can auto-justify its findings. Using such a 'monotonically increasing fitness chain' one can explore the degree to which the prediction system provides good predictions. If it does, then the candidate solutions in such a chain, should increasingly exhibit properties that ought to lead to a greater value of the predicted variable. In this way, SBSE can provide an alternative perspective on prediction system validation.

4. **Insight** Search based approaches can be used to find the so-called 'building blocks' of good solutions, rather than simply to find the solutions themselves. The building blocks of an optimal or near optimal solution are those sub parts of the solution that are shared by all good solutions. The identification of good building blocks can provide great insight into the solution space. For example, in predictive modeling, finding building blocks of good solutions may shed light on the relationship between clusters of related parameters that affect the predictive quality of a solution. This could potentially be far more revealing than principal component analysis, which it subsumes to some extent, because it reveals relationships between components of solutions, not merely the components themselves.

   Building blocks are an inherent feature of the workings of evolutionary approaches to optimization [28, 118]. However, they can be extracted from any search based approach, including local searchers [85]. Identifying the building blocks of a solution using search can be an end in itself, rather than a means to an end. It is not necessary to trust the algorithm to produce a solution, merely that it is able to identify building blocks likely to be worthy of consideration by the human; a far less stringent requirement. The ultimate solution proposed may be constructed by a human engineer and so the problem of acceptance simply evaporates. However, the engineer can have been informed and guided on the selection of parameters and construction of the model using a search based process that flags up potentially valuable building blocks.

8. HOW CAN PREDICTIVE MODELING HELP SBSE RESEARCHERS?

Section 7 explored the connections between predictive modeling and the SBSE community and, in particular, the strong connections between GP-based approaches to SBSE and predictive modeling. These connections suggest that work on predictive modeling may be as helpful to SBSE as SBSE is to predictive modeling. This section briefly explores the ways in which the results and techniques from the predictive modeling community may be useful to the SBSE research community[1]. The section briefly considers three areas of SBSE for which predictive models could be helpful and where there has been only a little initial work. This work demonstrates the need for predictive models but there remain many interesting open problems and challenges.

1. **Predictive Models as Fitness Functions** The two key ingredients for SBSE are the representation of candidate solutions and the fitness function [50]. The fitness function guides the search and so it is important that it correctly distinguishes good solutions from bad. We might say that we want the fitness function to be a good *predictor* of the attribute we seek to optimize. Naturally, the success of the whole SBSE enterprise rests upon finding fitness functions that are good predictors. In this way, there is a direct contribution to be made by the predictive modeling community.

   Defining suitable fitness functions is far from straightforward in many SBSE applications. For example, in Search Based Software Testing, there has been work on exploring the degree to which the fitness typically used is a good predictor [76].

   SBSE researchers also have to construct fitness functions that fit existing data (for simulations). The predictive modeling community has significant experience and expertise concerning missing and incomplete data that clearly should have a bearing on fitness function construction. Once we have a good prediction system, based on available data, we may think of this as a metric and, thereby, also think of it as a fitness function [54]. We may therefore claim, as we set out to do in the introduction that "Prediction Systems and Fitness Functions too".

2. **Predicting Global Optima Proximity** The SBSE approach reformulates a Software Engineering problem as a search problem. This implicitly creates a fitness landscape, like those depicted in Figure 1. Predictive Modeling may help to analyze and understand this landscape.

   For example, if the landscape consists of one smooth hill, then a simple hill climbing approach will find the global optimum. Gross et al. [47, 48] have explored the possibility of predicting the proximity of a result to a global optima for Search Based Software Testing. This is a very important topic, not just in SBSE research, but also in optimization in general. One of the advantages of greedy approaches for single objective test case selection and minimization [120] is the fact that greedy algorithms are known to be within a log of the global optimum.

3. **Effort prediction** Some searches using SBSE can consume considerable computation resources. While greedy algorithms and hill climbing tend to be fast and efficient (at the expense of often locating sub optimal solutions), more sophisticated search techniques such as evolutionary algorithms can be computationally intensive. This is important, since most of the existing work on SBSE has focused on the use of population based evolutionary algorithms, such as genetic algorithms and genetic programming [60].

   There is clearly scope for predictive modeling here. Predicting effort is one of the most widely studied aspects of predictive modeling, particularly with relation to software project cost and effort estimation [106]. Lammermann and Wegener have explored the possibility of predicting effort in Search Based Software Testing [77]. However, there is much more work that can be done on predictive models to inform SBSE.

   If effort can be predicted, then perhaps it can be controlled and thereby managed or even reduced. For example, there is work on transformation for search based testing, that seeks to reduce computation search effort by transforming programs for increased 'testability' [52, 56, 87]. This work is currently guided by known problems for testability. A more sophisticated approach might take account of predictions from models of testability. As was explained in Figure 1, different landscapes have very different properties, with some being more amenable to search based optimization. Therefore, the ability to predict features of the fitness landscape would be extremely attractive.

[1]The author is from the SBSE community and is not an expert in predictive modeling. As a result, this section is shorter than the remainder of the paper. However, this should not be interpreted as being anything other than a reflection of lack of expertise. An author with greater expertise in predictive modeling would surely produce a more detailed, authoritative and, no doubt, better account of the ways in which predictive modeling can be useful to SBSE researchers. Indeed, it is to be hoped that such a paper (or papers) may emerge from the predictive modeling community; it would be very welcome in the SBSE community.

9. CONCLUSIONS

This paper has explored some of the ways in which SBSE research may assist predictive modeling, outlining some open problems and challenges for the SBSE community. The paper also briefly reviews existing work on SBSE techniques for predictive modeling and how they may be further extended.

There is a close relationship between the challenges faced by both the SBSE and the Predictive Modeling communities. It seems likely that any research flow of ideas will be bi-directional. There are many ways in which the results and techniques emerging from the predictive modeling community may also be of benefit to researchers in SBSE. The connections between SBSE research work and predictive modeling outlined in this paper suggest that there may be significant value in a joint workshop to further explore these connections, relationships and to cross pollinate ideas between the two research communities.

10. ACKNOWLEDGEMENTS

The ideas presented here have been shaped by discussions with Giulio Antoniol, Edmund Burke, John Clark, Max Di Penta, Jose Javier Dolado, Norman Fenton, Robert Hierons, Mike Holcombe, Yue Jia, Brian Jones, Bill Langdon, Spiros Mancoridis, Phil McMinn, Tim Menzies, Marc Roper, Conor Ryan, Martin Shepperd, Paolo Tonella, Darrell Whitley, Xin Yao, Shin Yoo and Yuanyuan Zhang. I apologize to those I may have neglected to mention. My work is part funded by the generous support for grants from the EPSRC, the EU and Motorola.

11. REFERENCES

[1] W. Afzal, R. Torkar, and R. Feldt. Search-based prediction of fault count data. In *Proceedings of the 1st International Symposium on Search Based Software Engineering (SSBSE '09)*, pages 35–38, Cumberland Lodge, Windsor, UK, 13-15 May 2009. IEEE Computer Society.

[2] W. Afzal, R. Torkar, and R. Feldt. A systematic review of search-based testing for non-functional system properties. *Information and Software Technology*, 51(6):957–976, 2009.

[3] J. S. Aguilar-Ruiz, I. Ramos, J. C. Riquelme, and M. Toro. An Evolutionary Approach to Estimating Software Development Projects. *Information and Software Technology*, 43(14):875–882, December 2001.

[4] H. Akaike. A new look at the statistical model identification. *IEEE Transactions on Automatic Control*, 19(6):716–723, 1974.

[5] J. T. Alander, T. Mantere, and P. Turunen. Genetic Algorithm based Software Testing. In *Proceedings of the 3rd International Conference on Artificial Neural Networks and Genetic Algorithms (ICANNGA '97)*, pages 325–328, Norwich, UK, April 1997. Springer-Verlag.

[6] J. T. Alander, T. Mantere, P. Turunen, and J. Virolainen. GA in Program Testing. In *Proceedings of the 2nd Nordic Workshop on Genetic Algorithms and their Applications (2NWGA)*, pages 205–210, Vaasa, Finland, 19-23 August 1996.

[7] E. Alba and F. Chicano. Finding Safety Errors with ACO. In *Proceedings of the 9th annual Conference on Genetic and Evolutionary Computation (GECCO '07)*, pages 1066–1073, London, England, 7-11 July 2007. ACM.

[8] E. Alba and F. Chicano. Software Project Management with GAs. *Information Sciences*, 177(11):2380–2401, June 2007.

[9] S. Ali, L. C. Briand, H. Hemmati, and R. K. Panesar-Walawege. A systematic review of the application and empirical investigation of search-based test-case generation. *IEEE Transactions on Software Engineering*, 2010. To appear.

[10] G. Antoniol, M. Di Penta, and M. Harman. Search-based Techniques for Optimizing Software Project Resource Allocation. In *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation (GECCO '04)*, volume 3103/2004 of *Lecture Notes in Computer Science*, pages 1425–1426, Seattle, Washington, USA, 26-30 June 2004. Springer Berlin / Heidelberg.

[11] G. Antoniol, M. Di Penta, and M. Harman. Search-based Techniques Applied to Optimization of Project Planning for a Massive Maintenance Project. In *Proceedings of the 21st IEEE International Conference on Software Maintenance (ICSM '05)*, pages 240–249, Los Alamitos, California, USA, 25-30 September 2005. IEEE Computer Society.

[12] G. Antoniol, S. Gueorguiev, and M. Harman. Software project planning for robustness and completion time in the presence of uncertainty using multi objective search based software engineering. In *ACM Genetic and Evolutionary Computation COnference (GECCO 2009)*, pages 1673–1680, Montreal, Canada, 8th – 12th July 2009.

[13] F. Asadi, G. Antoniol, and Y. Guéhéneuc. Concept locations with genetic algorithms: A comparison of four distributed architectures. In *Proceedings of* 2^{nd} *International Symposium on Search based Software Engineering (SSBSE 2010)*, page To Appear, Benevento, Italy, 2010. IEEE Computer Society Press.

[14] A. J. Bagnall, V. J. Rayward-Smith, and I. M. Whittley. The Next Release Problem. *Information and Software Technology*, 43(14):883–890, December 2001.

[15] P. Baker, M. Harman, K. Steinhöfel, and A. Skaliotis. Search Based Approaches to Component Selection and Prioritization for the Next Release Problem. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM '06)*, pages 176–185, Philadelphia, Pennsylvania, 24-27 September 2006. IEEE Computer Society.

[16] A. Baresel, D. Binkley, M. Harman, and B. Korel. Evolutionary Testing in the Presence of Loop-Assigned Flags: A Testability Transformation Approach. In *Proceedings of the 2004 ACM SIGSOFT International Symposium on Software Testing and Analysis (ISSTA '04)*, pages 108–118, Boston, Massachusetts, USA, 11-14 July 2004. ACM.

[17] A. Baresel, H. Sthamer, and M. Schmidt. Fitness Function Design to Improve Evolutionary Structural Testing. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1329–1336, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[18] L. Bottaci. Instrumenting Programs with Flag Variables for Test Data Search by Genetic Algorithms. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1337–1342, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[19] S. Bouktif, D. Azar, D. Precup, H. Sahraoui, and B. Kégl. Improving Rule Set Based Software Quality Prediction: A Genetic Algorithm-based Approach. *Journal of Object Technology*, 3(4):227–241, 2004.

[20] S. Bouktif, B. Kégl, and H. Sahraoui. Combining Software Quality Predictive Models: An Evolutionary Approach. In *Proceedings of the International Conference on Software Maintenance (ICSM '02)*, pages 385–392, MontrÃl'al, Canada, 3-6 October 2002. IEEE Computer Society.

[21] S. Bouktif, H. Sahraoui, and G. Antoniol. Simulated Annealing for Improving Software Quality Prediction. In *Proceedings of the 8th annual Conference on Genetic and Evolutionary Computation (GECCO '06)*, pages 1893–1900, Seattle, Washington, USA, 8-12 July 2006. ACM.

[22] M. Bowman, L. Briand, and Y. Labiche. Multi-objective genetic algorithms to support class responsibility assignment. In 23^{rd} *IEEE International Conference on Software Maintenance (ICSM 2007)*, Los Alamitos, California, USA, October 2007. IEEE Computer Society Press. To Appear.

[23] L. C. Briand, Y. Labiche, and M. Shousha. Stress Testing Real-Time Systems with Genetic Algorithms. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1021–1028, Washington, D.C., USA, 25-29 June 2005. ACM.

[24] C. J. Burgess and M. Lefley. Can Genetic Programming Improve Software Effort Estimation? A Comparative Evaluation. *Information and Software Technology*, 43(14):863–873, December 2001.

[25] J. Clark, J. J. Dolado, M. Harman, R. M. Hierons, B. Jones,

M. Lumkin, B. Mitchell, S. Mancoridis, K. Rees, M. Roper, and M. Shepperd. Reformulating software engineering as a search problem. *IEE Proceedings — Software*, 150(3):161–175, 2003.

[26] S. L. Cornford, M. S. Feather, J. R. Dunphy, J. Salcedo, and T. Menzies. Optimizing Spacecraft Design - Optimization Engine Development: Progress and Plans. In *Proceedings of the IEEE Aerospace Conference*, pages 3681–3690, Big Sky, Montana, March 2003.

[27] O.-J. Dahl, E. W. Dijkstra, and C. A. R. Hoare. *Structured Programming*. Academic Press, London, 3 edition, 1972.

[28] K. De Jong. On using genetic algorithms to search program spaces. In J. J. Grefenstette, editor, *Genetic Algorithms and their Applications: Proceedings of the second international Conference on Genetic Algorithms*, pages 210–216, Hillsdale, NJ, USA, 28-31 July 1987. Lawrence Erlbaum Associates.

[29] J. T. de Souza, C. L. Maia, F. G. de Freitas, and D. P. Coutinho. The human competitiveness of search based software engineering. In *Proceedings of* 2^{nd} *International Symposium on Search based Software Engineering (SSBSE 2010)*, page To Appear, Benevento, Italy, 2010. IEEE Computer Society Press.

[30] C. Del Grosso, G. Antoniol, M. Di Penta, P. Galinier, and E. Merlo. Improving Network Applications Security: A New Heuristic to Generate Stress Testing Data. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1037–1043, Washington, D.C., USA, 25-29 June 2005. ACM.

[31] E. Díaz, J. Tuya, R. Blanco, and J. J. Dolado. A Tabu Search Algorithm for Structural Software Testing. *Computers & Operations Research*, 35(10):3052–3072, October 2008.

[32] J. J. Dolado. A Validation of the Component-based Method for Software Size Estimation. *IEEE Transactions on Software Engineering*, 26(10):1006–1021, October 2000.

[33] J. J. Dolado. On the Problem of the Software Cost Function. *Information and Software Technology*, 43(1):61–72, January 2001.

[34] J. J. Dolado and L. Fernandez. Genetic Programming, Neural Networks and Linear Regression in Software Project Estimation. In C. Hawkins, M. Ross, G. Staples, and J. B. Thompson, editors, *Proceedings of International Conference on Software Process Improvement, Research, Education and Training (INSPIRE III)*, pages 157–171, London, UK, 10-11 September 1998. British Computer Society.

[35] M. D. Ernst, J. Cockrell, W. G. Griswold, and D. Notkin. Dynamically discovering likely program invariants to support program evolution. *IEEE Transactions on Software Engineering*, 27(2):1–25, Feb. 2001.

[36] R. M. Everson and J. E. Fieldsend. Multiobjective Optimization of Safety Related Systems: An Application to Short-Term Conflict Alert. *IEEE Transactions on Evolutionary Computation*, 10(2):187–198, April 2006.

[37] M. P. Evett, T. M. Khoshgoftaar, P. der Chien, and E. B. Allen. Using Genetic Programming to Determine Software Quality. In *Proceedings of the 12th International Florida Artificial Intelligence Research Society Conference (FLAIRS '99)*, pages 113–117, Orlando, FL, USA, 3-5 May 1999. Florida Research Society.

[38] M. S. Feather, S. L. Cornford, J. D. Kiper, and T. Menzies. Experiences using Visualization Techniques to Present Requirements, Risks to Them, and Options for Risk Mitigation. In *Proceedings of the International Workshop on Requirements Engineering Visualization (REV '06)*, pages 10–10, Minnesota, USA, 11 September 2006. IEEE.

[39] M. S. Feather and T. Menzies. Converging on the Optimal Attainment of Requirements. In *Proceedings of the 10th IEEE International Conference on Requirements Engineering (RE '02)*, pages 263–270, Essen, Germany, 9-13 September 2002. IEEE.

[40] N. E. Fenton and M. Neil. A critique of software defect prediction models. *IEEE Transactions on Software Engineering*, 25(5):675–689, 1999.

[41] A. Finkelstein, M. Harman, S. A. Mansouri, J. Ren, and Y. Zhang. "Fairness Analysis" in Requirements Assignments. In *Proceedings of the 16th IEEE International Requirements Engineering Conference (RE '08)*, pages 115–124, Barcelona, Catalunya, Spain, 8-12 September 2008. IEEE Computer Society.

[42] R. W. Floyd. Assigning meanings to programs. In J. T. Schwartz, editor, *Mathematical Aspects of Computer Science*, volume 19 of *Symposia in Applied Mathematics*, pages 19–32. American Mathematical Society, Providence, RI, 1967.

[43] T. Foss, E. Stensrud, B. Kitchenham, and I. Myrtveit. A simulation study of the model evaluation criterion mmre. *IEEE Transactions on Software Engineering*, 29:985–995, 2003.

[44] V. Garousi, L. C. Briand, and Y. Labiche. Traffic-aware Stress Testing of Distributed Real-Time Systems based on UML Models using Genetic Algorithms. *Journal of Systems and Software*, 81(2):161–185, February 2008.

[45] G. Gay, T. Menzies, M. Davies, and K. Gundy-Burlet. Automatically finding the control variables for complex system behavior. *Automated Software Engineering*, page to appear.

[46] N. Gold, M. Harman, Z. Li, and K. Mahdavi. Allowing Overlapping Boundaries in Source Code using a Search Based Approach to Concept Binding. In *Proceedings of the 22nd IEEE International Conference on Software Maintenance (ICSM '06)*, pages 310–319, Philadelphia, USA, 24-27 September 2006. IEEE Computer Society.

[47] H.-G. Groß, B. F. Jones, and D. E. Eyres. Structural Performance Measure of Evolutionary Testing applied to Worst-Case Timing of Real-Time Systems. *IEE Proceedings - Software*, 147(2):25–30, April 2000.

[48] H.-G. Groß and N. Mayer. Evolutionary Testing in Component-based Real-Time System Construction. In E. Cantú-Paz, editor, *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 207–214, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[49] R. J. Hall. A quantum algorithm for software engineering search. In *Automated Software Engineering (ASE 2009)*, pages 40–51, Auckland, New Zealand, 2009. IEEE Computer Society.

[50] M. Harman. The current state and future of search based software engineering. In L. Briand and A. Wolf, editors, *Future of Software Engineering 2007*, pages 342–357, Los Alamitos, California, USA, 2007. IEEE Computer Society Press.

[51] M. Harman. Search based software engineering for program comprehension. In 15^{th} *International Conference on Program Comprehension (ICPC 07)*, pages 3–13, Banff, Canada, 2007. IEEE Computer Society Press.

[52] M. Harman. Open problems in testability transformation (keynote). In *1st International Workshop on Search Based Testing (SBT 2008)*, Lillehammer, Norway, 2008.

[53] M. Harman. Why the virtual nature of software makes it ideal for search based optimization. In 13^{th} *International Conference on Fundamental Approaches to Software Engineering (FASE 2010)*, pages 1–12, Paphos, Cyprus, March 2010.

[54] M. Harman and J. A. Clark. Metrics Are Fitness Functions Too. In *Proceedings of the 10th IEEE International Symposium on Software Metrics (METRICS '04)*, pages 58–69, Chicago, USA, 11-17 September 2004. IEEE Computer Society.

[55] M. Harman, R. Hierons, and M. Proctor. A New Representation and Crossover Operator for Search-based Optimization of Software Modularization. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1351–1358, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[56] M. Harman, L. Hu, R. M. Hierons, J. Wegener, H. Sthamer, A. Baresel, and M. Roper. Testability Transformation. *IEEE Transactions on Software Engineering*, 30(1):3–16, January 2004.

[57] M. Harman and B. F. Jones. Search based software engineering. *Information and Software Technology*, 43(14):833–839, Dec. 2001.

[58] M. Harman, J. Krinke, J. Ren, and S. Yoo. Search based data sensitivity analysis applied to requirement engineering. In *ACM Genetic and Evolutionary Computation COnference (GECCO 2009)*, pages 1681–1688, Montreal, Canada, 8th – 12th July 2009.

[59] M. Harman, K. Lakhotia, and P. McMinn. A multi-objective approach to search-based test data generation. In *GECCO 2007: Proceedings of the* 9^{th} *annual conference on Genetic and evolutionary computation*, pages 1098 – 1105, London, UK, July 2007. ACM Press.

[60] M. Harman, A. Mansouri, and Y. Zhang. Search based software engineering: A comprehensive analysis and review of trends techniques and applications. Technical Report TR-09-03, Department of Computer Science, King's College London, April 2009.

[61] M. Harman and L. Tratt. Pareto optimal search-based refactoring at the design level. In *GECCO 2007: Proceedings of the* 9^{th} *annual conference on Genetic and evolutionary computation*, pages 1106 – 1113, London, UK, July 2007. ACM Press.

[62] C. A. R. Hoare. An Axiomatic Basis of Computer Programming. *Communications of the ACM*, 12:576–580, 1969.

[63] C. A. R. Hoare. How did software get so reliable without proof? In *IEEE International Conference on Software Engineering (ICSE'96)*, Los Alamitos, California, USA, 1996. IEEE Computer Society Press. Keynote talk.

[64] A. Idri, T. M. Khoshgoftaar, and A. Abran. Can neural networks be easily interpreted in software cost estimation?, 2003.

[65] D. Jackson. Self-adaptive focusing of evolutionary effort in hierarchical genetic programming. In A. Tyrrell, editor, *2009 IEEE Congress on Evolutionary Computation*, pages –, Trondheim, Norway, 18-21 May 2009. IEEE Computational Intelligence Society, IEEE Press.

[66] O. Jalali, T. Menzies, and M. Feather. Optimizing Requirements Decisions With KEYS. In *Proceedings of the 4th International Workshop on Predictor Models in Software Engineering (PROMISE '08)*, pages 79–86, Leipzig, Germany, 12-13 May 2008. ACM.

[67] G. Jarillo, G. Succi, W. Pedrycz, and M. Reformat. Analysis of Software Engineering Data using Computational Intelligence Techniques. In *Proceedings of the 7th International Conference on Object Oriented Information Systems (OOIS '01)*, pages 133–142, Calgary, Canada, 27-29 August 2001. Springer.

[68] A. M. Joshi, L. Eeckhout, L. K. John, and C. Isen. Automated Microprocessor Stressmark Generation. In *Proceedings of the 14th IEEE International Symposium on High Performance Computer Architecture (HPCA '08)*, pages 229–239, Salt Lake City, UT, USA, 16-20 February 2008. IEEE.

[69] T. M. Khoshgoftaar and Y. Liu. A Multi-Objective Software Quality Classification Model Using Genetic Programming. *IEEE Transactions on Reliability*, 56(2):237–245, June 2007.

[70] T. M. Khoshgoftaar, Y. Liu, and N. Seliya. Genetic Programming-based Decision Trees for Software Quality Classification. In *Proceedings of the 15th International Conference on Tools with Artificial Intelligence (ICTAI '03)*, pages 374–383, Sacramento, California, USA, 3-5 November 2003. IEEE Computer Society.

[71] T. M. Khoshgoftaar, N. Seliya, and D. J. Drown. On the Rarity of Fault-prone Modules in Knowledge-based Software Quality Modeling. In *Proceedings of the 20th International Conference on Software Engineering and Knowledge Engineering (SEKE '08)*, pages 279–284, San Francisco, CA, USA, 1-3 July 2008. Knowledge Systems Institute Graduate School.

[72] T. M. Khoshgoftaar, L. Yi, and N. Seliya. A multiobjective module-order model for software quality enhancement. *IEEE Transactions on Evolutionary Computation*, 8(6):593– 608, December 2004.

[73] C. Kirsopp, M. Shepperd, and J. Hart. Search heuristics, case-based reasoning and software project effort prediction. In *GECCO 2002: Proceedings of the Genetic and Evolutionary Computation Conference*, pages 1367–1374, San Francisco, CA 94104, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[74] J. R. Koza. *Genetic Programming: On the Programming of Computers by Means of Natural Selection*. MIT Press, Cambridge, MA, 1992.

[75] M. Kuperberg, K. Krogmann, and R. Reussner. Performance Prediction for Black-Box Components Using Reengineered Parametric Behaviour Models. In *Proceedings of the 11th International Symposium on Component-Based Software Engineering (CBSE '08)*, volume 5282 of *LNCS*, pages 48–63, Karlsruhe, Germany, 14-17 October 2008. Springer.

[76] F. Lammermann, A. Baresel, and J. Wegener. Evaluating Evolutionary Testability with Software-Measurements. In *Proceedings of the 2004 Conference on Genetic and Evolutionary Computation (GECCO '04)*, volume 3103 of *Lecture Notes in Computer Science*, pages 1350–1362,

Seattle, Washington, USA, 26-30 June 2004. Springer Berlin / Heidelberg.

[77] F. Lammermann and S. Wappler. Benefits of Software Measures for Evolutionary White-Box Testing. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1083–1084, Washington, D.C., USA, 25-29 June 2005. ACM.

[78] W. B. Langdon, M. Harman, and Y. Jia. Multi objective mutation testing with genetic programming. In 4^{th} *Testing Academia and Industry Conference — Practice And Research Techniques (TAIC PART'09)*, pages 21–29, Windsor, UK, 4th–6th September 2009.

[79] W. B. Langdon and R. Poli. *Foundations of Genetic Programming*. Springer-Verlag, 2002.

[80] M. Lefley and M. J. Shepperd. Using Genetic Programming to Improve Software Effort Estimation Based on General Data Sets. In *Proceedings of the 2003 Conference on Genetic and Evolutionary Computation (GECCO '03)*, volume 2724 of *LNCS*, pages 2477–2487, Chicago, Illinois, USA, 12-16 July 2003. Springer.

[81] Z. Li, M. Harman, and R. M. Hierons. Search Algorithms for Regression Test Case Prioritization. *IEEE Transactions on Software Engineering*, 33(4):225–237, April 2007.

[82] Y. Liu and T. Khoshgoftaar. Reducing Overfitting in Genetic Programming Models for Software Quality Classification. In *Proceedings of the 8th IEEE International Symposium on High Assurance Systems Engineering (HASE '04)*, pages 56–65, Tampa, Florida, USA, 25-26 March 2004. IEEE Computer Society.

[83] Y. Liu and T. M. Khoshgoftaar. Genetic Programming Model for Software Quality Classification. In *Proceedings of the 6th IEEE International Symposium on High-Assurance Systems Engineering: Special Topic: Impact of Networking (HASE '01)*, pages 127–136, Boco Raton, FL, USA, 22-24 October 2001. IEEE Computer Society.

[84] Y. Liu and T. M. Khoshgoftaar. Building Decision Tree Software Quality Classification Models Using Genetic Programming. In *Proccedings of the Genetic and Evolutionary Computation Conference (GECCO '03)*, volume 2724 of *LNCS*, pages 1808–1809, Chicago, Illinois, USA, 12-16 July 2003. Springer.

[85] K. Mahdavi, M. Harman, and R. M. Hierons. A multiple hill climbing approach to software module clustering. In *IEEE International Conference on Software Maintenance*, pages 315–324, Los Alamitos, California, USA, Sept. 2003. IEEE Computer Society Press.

[86] P. McMinn. Search-based Software Test Data Generation: A Survey. *Software Testing, Verification and Reliability*, 14(2):105–156, 2004.

[87] P. McMinn. Search-based failure discovery using testability transformations to generate pseudo-oracles. In F. Rothlauf, editor, *Genetic and Evolutionary Computation Conference (GECCO 2009)*, pages 1689–1696, Montreal, Québec, Canada, 2009. ACM.

[88] P. McMinn, D. Binkley, and M. Harman. Testability transformation for efficient automated test data search in the presence of nesting. In *UK Software Testing Workshop (UK Test 2005)*, Sheffield, UK, Sept. 2005.

[89] P. McMinn, D. Binkley, and M. Harman. Empirical evaluation of a nesting testability transformation for evolutionary testing. *ACM Transactions on Software Engineering and Methodology*, 18(3), May 2009. Article 11.

[90] P. McMinn, M. Harman, D. Binkley, and P. Tonella. The Species per Path Approach to Search-based Test Data Generation. In *Proceedings of the 2006 International Symposium on Software Testing and Analysis (ISSTA '06)*, pages 13–24, Portland, Maine, USA., 17-20 July 2006. ACM.

[91] P. McMinn and M. Holcombe. The State Problem for Evolutionary Testing. In *Proceedings of the 2003 Conference on Genetic and Evolutionary Computation (GECCO '03)*, volume 2724 of *LNCS*, pages 2488–2498, Chicago, Illinois, USA, 12-16 July 2003. Springer.

[92] B. S. Mitchell and S. Mancoridis. Using Heuristic Search Techniques to Extract Design Abstractions from Source Code. In *Proceedings of the 2002 Conference on Genetic and Evolutionary Computation (GECCO '02)*, pages 1375–1382, New York, USA, 9-13 July 2002. Morgan Kaufmann Publishers.

[93] B. S. Mitchell and S. Mancoridis. On the Automatic Modularization of Software Systems using the Bunch Tool. *IEEE Transactions on Software Engineering*, 32(3):193–208, March 2006.

[94] B. S. Mitchell, M. Traverso, and S. Mancoridis. An architecture for distributing the computation of software clustering algorithms. In *IEEE/IFIP Proceedings of the Working Conference on Software Architecture (WICSA '01)*, pages 181–190, Amsterdam, Netherlands, 2001. IEEE Computer Society.

[95] M. O'Keeffe and M. Ó Cinnéide. Search-based Software Maintenance. In *Proceedings of the Conference on Software Maintenance and Reengineering (CSMR '06)*, pages 249–260, Bari, Italy, 22-24 March 2006. IEEE Computer Society.

[96] T. J. Ostrand, E. J. Weyuker, and R. M. Bell. Predicting the location and number of faults in large software systems. *IEEE Transactions on Software Engineering*, 31(4):340–355, 2005.

[97] M. Pelikan, D. E. Goldberg, and E. Cantú-Paz. BOA: The Bayesian Optimization Algorithm. Technical Report IlliGAL 99003, Illinois Genetic Algorithms Laboratory, University of Illinois at Urbana-Champaign, Urbana, IL, 1999.

[98] K. Praditwong, M. Harman, and X. Yao. Software module clustering as a multi-objective search problem. *IEEE Transactions on Software Engineering*, 2010. to appear.

[99] R. Pressman. *Software Engineering: A Practitioner's Approach*. McGraw-Hill Book Company Europe, Maidenhead, Berkshire, England, UK., 3rd edition, 1992. European adaptation (1994). Adapted by Darrel Ince. ISBN 0-07-707936-1.

[100] O. Räihä. A survey on search based software design. Technical Report Technical Report D-2009-1, Department of Computer Sciences, University of Tampere, 2009.

[101] A. Ratle. Accelerating the convergence of evolutionary algorithms by fitness landscape approximation. In A. E. Eiben, T. Back, M. Schoenauer, and H.-P. Schwefel, editors, 5^{th} *Parallel Problem Solving from Nature (PPSN'98)*, volume 1498 of *Lecture Notes in Computer Science (LNCS)*, pages 87–96. Springer-Verlag (New York), Amsterdam, The Netherlands, Sept. 1998.

[102] M. O. Saliu and G. Ruhe. Bi-objective release planning for evolving software systems. In I. Crnkovic and A. Bertolino, editors, *Proceedings of the 6^{th} joint meeting of the European Software Engineering Conference and the ACM SIGSOFT International Symposium on Foundations of Software Engineering (ESEC/FSE) 2007*, pages 105–114. ACM, Sept. 2007.

[103] G. E. Schwartz. Estimating the dimension of a model. *Annals of Mathematical Statistics*, 6:461–464, 1978.

[104] O. Seng, M. Bauer, M. Biehl, and G. Pache. Search-based Improvement of Subsystem Decompositions. In *Proceedings of the 2005 Conference on Genetic and Evolutionary Computation (GECCO '05)*, pages 1045–1051, Washington, D.C., USA, 25-29 June 2005. ACM.

[105] Y. Shan, R. I. McKay, C. J. Lokan, and D. L. Essam. Software Project Effort Estimation Using Genetic Programming. In *Proceedings of the 2002 IEEE International Conference on Communications, Circuits and Systems and West Sino Expositions (ICCCSWSE '02)*, volume 2, pages 1108–1112, Chengdu, China, 29 June-1 July 2002. IEEE.

[106] M. Shepperd. Software economics. In L. Briand and A. Wolf, editors, *Future of Software Engineering 2007*, pages 304–315, Los Alamitos, California, USA, 2007. IEEE Computer Society Press.

[107] M. J. Shepperd. *Foundations of software measurement*. Prentice Hall, 1995.

[108] A. F. Sheta. Estimation of the COCOMO Model Parameters Using Genetic Algorithms for NASA Software Projects. *Journal of Computer Science*, 2(2):118–123, 2006.

[109] K. K. Shukla. Neuro-Genetic Prediction of Software Development Effort. *Information and Software Technology*, 42(10):701–713, July 2000.

[110] C. L. Simons and I. C. Parmee. Agent-based Support for Interactive Search in Conceptual Software Engineering Design. In M. Keijzer, editor, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 1785–1786, Atlanta, GA, USA, 12-16 July 2008. ACM.

[111] I. Sommerville. *Software Engineering*. Addison-Wesley, 6th edition, 2001.

[112] R. Vivanco and D. Jin. Enhancing Predictive Models using Principal Component Analysis and Search Based Metric Selection: A Comparative Study. In *Proceedings of the Second ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM '08)*, pages 273–275, Kaiserslautern, Germany, 9-10 October 2008. ACM.

[113] J. Wegener, A. Baresel, and H. Sthamer. Evolutionary Test Environment for Automatic Structural Testing. *Information and Software Technology Special Issue on Software Engineering using Metaheuristic Innovative Algorithms*, 43(14):841–854, December 2001.

[114] J. Wegener and F. Mueller. A Comparison of Static Analysis and Evolutionary Testing for the Verification of Timing Constraints. *Real-Time Systems*, 21(3):241–268, November 2001.

[115] W. Weimer, T. V. Nguyen, C. L. Goues, and S. Forrest. Automatically finding patches using genetic programming. In *International Conference on Software Engineerign (ICSE 2009)*, pages 364–374, Vancouver, Canada, 2009.

[116] D. R. White, J. A. Clark, J. Jacob, and S. M. Poulding. Searching for Resource-Efficient Programs: Low-Power Pseudorandom Number Generators. In M. Keijzer, editor, *Proceedings of the 10th Annual Conference on Genetic and Evolutionary Computation (GECCO '08)*, pages 1775–1782, Atlanta, GA, USA, 12-16 July 2008. ACM.

[117] G. Winter, J. Periaux, M. Galan, and P. Cuesta. *Genetic Algorithms in Engineering and Computer Science*. Wiley, 1995.

[118] A. S. Wu and R. K. Lindsay. A comparison of the fixed and floating building block representation in the genetic algorithm. *Evolutionary Computation*, 4(2):169–193, 1997.

[119] S. Yoo and M. Harman. Regression testing minimisation, selection and prioritisation: A survey. *Journal of Software Testing, Verification and Reliability*. to appear.

[120] S. Yoo and M. Harman. Pareto efficient multi-objective test case selection. In *International Symposium on Software Testing and Analysis (ISSTA'07)*, pages 140 – 150, London, United Kingdom, July 2007. Association for Computer Machinery.

[121] B.-T. Zhang. A bayesian framework for evolutionary computation. In *1999 Congress on Evolutionary Computation*, pages 722–728, Piscataway, NJ, 1999. IEEE.

[122] Y. Zhang, A. Finkelstein, and M. Harman. Search based requirements optimisation: Existing work and challenges. In *International Working Conference on Requirements Engineering: Foundation for Software Quality (REFSQ'08)*, volume 5025, pages 88–94, Montpellier, France, 2008. Springer LNCS.

[123] Y. Zhang, M. Harman, and A. Mansouri. The multi-objective next release problem. In *GECCO 2007: Proceedings of the 9^{th} annual conference on Genetic and evolutionary computation*, pages 1129 – 1137, London, UK, July 2007. ACM Press.