Search Based Software Engineering

Mark Micallef

Department of Computer Science, University of Malta
mark.micallef@um.edu.mt

1 Motivation

Consider the following questions, which are posed by software engineers on a daily basis:

1. What is the smallest set of test cases that will cover all statements in this program?
2. What is the best way to organise classes and methods for this OO design?
3. What is the set of requirements that balances software development cost and customer satisfaction?

Whilst these questions seem to be addressing different problems, they do have some notable commonalities. Firstly, they form part of a large set of software engineering problems which can each be solved by a multitude of potential solutions. That is to say that if one were to ask the above questions to x equally competent engineers, one would likely get back x different yet correct solutions. Secondly, this class of problems is usually tasked with balancing a number of competing constraints. A typical example here is maximising customer satisfaction whilst keeping development costs low. Finally, whilst there is typically no perfect answer (and indeed no precise rules for computing the best solution), good solutions can be recognised.

When problems with similar characteristics were encountered in disciplines other than software engineering, they were solved with a large degree of success using search-based techniques. It was this realisation that gave rise to the field of search based software engineering.

2 Search Based Software Engineering

Search Based Software Engineering (SBSE) is the name given to a body of work in which Search Based Optimisation is applied to Software Engineering problems. Although the literature reveals earlier work in the area, the term itself was coined by Harman and Jones [1] in 2001, an event which seems to have legitimised the area as a sub-field of software engineering and led to an explosion of interest in the area.

Attempting to solve a problem using these techniques essentially requires reformulating software engineering as a search problem. That is to say that for a given problem, one needs to define:

– a representation of the problem which is conducive to symbolic manipulation
– a fitness function defined in terms of this representation

Consider the question *"what is the smallest set of test cases that will cover all statements in this program?"*. Assuming for the sake of example that the program is a method which takes two integers as parameters, one could propose an encoding whereby test cases are encoded as ordered pairs $\langle x, y \rangle$ where $x, y \in \mathbb{Z}$. Furthermore, the fitness function can be defined as the function which takes a set of integer pairs and returns a measure of coverage between 0 and 1.

$$f : \mathbb{P}(\mathbb{Z} \times \mathbb{Z}) \to \mathbb{R}$$

The problem is thus reformulated as a search problem whereby the goal is to find the smallest list of integer pairs with a fitness function of 1.

Whilst the example is a simple one, it demonstrates that once you define an encoding and a fitness function, it simply becomes a matter of applying known techniques from search-based optimisation to find a solution to your problem. Of course, defining an encoding and fitness function is not always straight forward. In most non-trivial scenarios, one needs to deal with multiple goals to a fitness function, conflicting goals, and ones which are not easily encodable in an objective manner.

3 Talk outline

During my talk at the CSAW workshop, I will introduce the topic, discuss commonly used algorithms in this field and provide two or three concrete examples to illustrate what can be achieved using these techniques. I will then outline ongoing personal research work in this particular field.

References

1. Mark Harman and Bryan F Jones. Search-based software engineering. *Information and Software Technology*, 43(14):833–839, 2001.