Object-Oriented Evolutionary Testing: A Review of Evolutionary Approaches to the Generation of Test Data for Object-Oriented Software

Ana Filipa Nogueira, University of Coimbra, Coimbra, Portugal
José Carlos Bregieiro Ribeiro, Polytechnic Institute of Leiria, Leiria, Portugal
Francisco Fernández de Vega, University of Extremadura, Mérida, Spain
Mário Alberto Zenha-Rela, University of Coimbra, Coimbra, Portugal

ABSTRACT

In Object-Oriented Evolutionary Testing, metaheuristics are employed to select or generate Test Data for Object-Oriented software. The application of search-based strategies to the Software Testing of Object-Oriented Software is fairly recent and is yet to be investigated comprehensively; this article aims to explore, review and contextualize relevant literature and research in this area, while pinpointing open problems and setting grounds for future work.

Keywords: Evolutionary Algorithms, Evolutionary Testing, Object-Orientation, Search-Based Software Engineering, Software Testing

INTRODUCTION

Search-Based Software Engineering (SBSE) seeks to reformulate Software Engineering (SE) problems as search-based optimisation problems. It has been applied to a wide variety of SE areas, including requirements engineering, project planning and cost estimation, automated maintenance, service-oriented software engineering, compiler optimisation and quality assessment (Harman, 2007). Most of the overall literature (an estimated 59%) in the SBSE area is, however, concerned with Software Testing (ST) related applications, with structural test data generation being the most studied sub-topic (Harman, Mansouri, & Zhang, 2009).

The application of Evolutionary Algorithms (EAs) to test data generation or selection is often referred to as Evolutionary Testing (ET) (Tonella, 2004b; Wappler & Wegener, 2006b) or
Search-Based Test Data Generation (SBTDG) (McMinn, 2004). ET consists of exploring the space of test programs by using metaheuristic techniques that direct the search towards the potentially most promising areas of the input space (Bertolino, 2007); its foremost objective is usually that of searching for a set of test programs that satisfies a predefined test criterion.

EAs have already been applied, with significant success, to the search for test data; the first application of heuristic optimisation techniques to test data generation was presented in 1992 (Xanthakis, Ellis, Skourlas, Gall, & K. Karapoulios, 1992). However, research has been mainly geared towards generating test data for procedural software, and traditional methods – despite their effectiveness and efficiency – cannot be applied without adaptation to Object-Oriented (OO) systems.

The application of search-based strategies to unit testing of OO programs is, in fact, fairly recent – the first approach was presented in 2004 (Tonella, 2004b) – and is yet to be investigated comprehensively (Harman, Has-soun, Lakhota, McMinn, & Wegener, 2007). Interesting review articles on the topic of SBSE, and Search-Based Software Testing (SBST) in particular, include: (McMinn, 2004; Mantere & Alander, 2005; Xiao, El-Attar, Reformat & Miller, 2007; Afzal, Torkar, & Feldt, 2009; Ali, Briand, Hemmati, & Panesar-Walawege, 2009; Harman et al., 2009; Harman & McMinn, 2010; Maragathavalli, 2011; McMinn, 2011; Harman, Mansouri, & Zhang, 2012; Varshney & Mehrotra, 2013; Anand et al., 2013). McMinn surveys the use of metaheuristic search techniques for the automatic generation of test data (McMinn, 2004); because the work on SBST had, thus far (2004), been largely restricted to programs of a procedural nature, these are the main subject of this review. In (Mantere & Alander, 2005), a review of the application of Genetic Algorithm (GA)-based optimisation methods to ST is presented; the authors stress out that all the researchers in this area report good (or, at least, encouraging) results regarding their use. Xiao et al. reported the experimental results regarding the effectiveness of five different optimisation techniques over five different C/C++ programs (Xiao et al., 2007). The results show that the GA-based approach outperformed the remaining techniques – e.g., Simulated Annealing (SA), Genetic Simulated Annealing (GSA), Simulated Annealing with Advanced Adaptive Neighborhood (SA/ANN) and Random Testing – achieving the best overall performance.

The work proposed in (Afzal et al., 2009) is built on McMinn’s research (McMinn, 2004) and presented a review on how search-based techniques are used to test non-functional properties of the software, focusing on: the properties studied, the fitness functions implemented and the constraints and limitations found when testing each property. The set of non-functional properties identified include (listed in descending order of the number of papers that investigate it): execution time, security, usability, safety and quality of service. A systematic review on the way SBST techniques have been empirically assessed is presented in (Ali et al., 2009). In (Harman et al., 2009, 2012) a thorough index and classification of SBSE-related literature is provided, supported by an online repository. Local search, SA, GAs and Genetic Programming (GP) are identified as the most widely used optimisation and search techniques. Harman and McMinn conducted a theoretical and empirical study on the SBST field with the purposes of predicting the scenarios in which ET should perform properly and of justifying the reasons why a specific technique suited a particular situation (Harman & McMinn, 2010). The predictions were supported by empirical studies which showed that sometimes a simpler solution can be more suitable than a more sophisticated one, and it was theoretically and empirically proved that an evolutionary approach is suitable for several scenarios.

Maragathavalli also performed an overview of the current SBST techniques, and pointed out that for programs in which the complexity of the input domain grows, the efficacy of GA is quite significant when compared to random testing (Maragathavalli, 2011). (Varshney & Mehrotra, 2013) overviews the SBST techniques
Autonomously Evolving Communication Protocols: The Case of Multi-Hop Broadcast
www.igi-global.com/chapter/autonomously-evolving-communication-protocols/58307?camid=4v1a