The Oracle Problem in Software Testing: A Survey

Earl T. Barr, Mark Harman, Phil McMinn, Muzammil Shahbaz, and Shin Yoo

Abstract—Testing involves examining the behaviour of a system in order to discover potential faults. Given an input for a system, the challenge of distinguishing the corresponding desired, correct behaviour from potentially incorrect behavior is called the "test oracle problem". Test oracle automation is important to remove a current bottleneck that inhibits greater overall test automation. Without test oracle automation, the human has to determine whether observed behaviour is correct. The literature on test oracles has introduced techniques for oracle automation, including modelling, specifications, contract-driven development and metamorphic testing. When none of these is completely adequate, the final source of test oracle information remains the human, who may be aware of informal specifications, expectations, norms and domain specific information that provide informal oracle guidance. All forms of test oracles, even the humble human, involve challenges of reducing cost and increasing benefit. This paper provides a comprehensive survey of current approaches to the test oracle problem and an analysis of trends in this important area of software testing research and practice.

Index Terms—Test oracle, automatic testing, testing formalism

1 Introduction

Much work on software testing seeks to automate as much of the test process as practical and desirable, to make testing faster, cheaper, and more reliable. To this end, we need a test oracle, a procedure that distinguishes between the correct and incorrect behaviors of the System Under Test (SUT).

However, compared to many aspects of test automation, the problem of automating the test oracle has received significantly less attention, and remains comparatively less well-solved. This current open problem represents a significant bottleneck that inhibits greater test automation and uptake of automated testing methods and tools more widely. For instance, the problem of automatically generating test inputs has been the subject of research interest for nearly four decades [46], [107]. It involves finding inputs that cause execution to reveal faults, if they are present, and to give confidence in their absence, if none are found. Automated test input generation been the subject of many significant advances in both Search-Based Testing [3], [5], [83], [126], [128] and Dynamic Symbolic Execution [75], [108], [161]; yet none of these advances address the issue of checking generated inputs with respect to expected behaviours—that is, providing an automated solution to the test oracle problem.

Of course, one might hope that the SUT has been developed under excellent design-for-test principles, so that there might be a detailed, and possibly formal, specification of intended behaviour. One might also hope that the code itself contains pre- and post- conditions that implement well-understood contract-driven development approaches [135]. In these situations, the test oracle cost problem is ameliorated by the presence of an automatable test oracle to which a testing tool can refer to check outputs, free from the need for costly human intervention.

Where no full specification of the properties of the SUT exists, one may hope to construct a partial test oracle that can answer questions for some inputs. Such partial test oracles can be constructed using metamorphic testing (built from known relationships between desired behaviour) or by deriving oracular information from execution or documentation.

For many systems and most testing as currently practiced in industry, however, the tester does not have the luxury of formal specifications or assertions, or automated partial test oracles [91], [92]. The tester therefore faces the daunting task of manually checking the system's behaviour for all test cases. In such cases, automated software testing approaches must address the human oracle cost problem [1], [82], [130].

To achieve greater test automation and wider uptake of automated testing, we therefore need a concerted effort to find ways to address the test oracle problem and to integrate automated and partially automated test oracle solutions into testing techniques. This paper seeks to help address this challenge by providing a comprehensive review and analysis of the existing literature of the test oracle problem.

Four partial surveys of topics relating to test oracles precede this one. However, none has provided a comprehensive survey of trends and results. In 2001, Baresi and Young [17] presented a partial survey that covered four topics prevalent at the time the paper was published: assertions, specifications, state-based conformance testing, and log file analysis. While these topics remain important, they capture only a part of the overall landscape of research in test

- *E.T. Barr, M. Harman and S. Yoo are with the Department of Computer Science, University College London, Gower Street, London WC2R 2LS, London, United Kingdom. E-mail: {e.barr, m.harman, shin.yoo}@ucl.ac.uk.*
- *P. McMinn and M. Shahbaz are with the Department of Computer Science, University of Sheffield, Sheffield S1 4DP, South Yorkshire, United Kingdom.*
  E-mail: p.mcminn@sheffield.ac.uk, muzammil.shahbaz@gmail.com.

Manuscript received 24 July 2013; revised 5 Sept. 2014; accepted 28 Oct. 2014. Date of publication 19 Nov. 2014; date of current version 15 May 2015. Recommended for acceptance by L. Baresi.
For information on obtaining reprints of this article, please send e-mail to: reprints@ieee.org, and reference the Digital Object Identifier below.
Digital Object Identifier no. 10.1109/TSE.2014.2372785