Information and Software Technology 78 (2016) 1–26

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

An incremental method for extracting tests from object-oriented specification

M. Ghoreshi, H. Haghighi*

Faculty of Computer Science and Engineering, Shahid Beheshti University G. C., Tehran, Iran

ARTICLE INFO

Article history:
Received 4 January 2016
Revised 11 May 2016
Accepted 13 May 2016
Available online 26 May 2016

Keywords:
Test case generation
Formal specification
Specification-based testing
Object orientation
Object-Z

ABSTRACT

Context: The nature of the object-oriented development process is iterative and incremental, and through this process, software artifacts are refined and evolved continuously; however, most of proposed methods for deriving test cases from formal, object-oriented specifications have been adapted from previous structural techniques and are not aligned with such an incremental process. These methods are not adaptive with changes in the software specification, and there is no mechanism to evolve test artifacts respectively. Moreover, the existing methods do not cover all different object-oriented testing levels, i.e., intra-method, inter-method, intra-class and inter-class levels.

Objective: This paper presents an incremental method for extracting tests from formal, object-oriented specifications. Extracted tests are adaptive with changes in the class specification. In addition, the proposed method covers all different object-oriented testing levels.

Method: We first make a test machine (as a new notion introduced in this paper) for each class operation to cover the intra-method test level. With the combination of these test machines, new test machines covering the inter-method and intra-class test levels can be made. Extracted tests can be easily modified when the class specification is modified, and, in this way, our approach enables iterative and incremental test derivation. With test machines corresponding to a class hierarchy, this approach can also be used for deriving inter-class tests.

Results: As a case study, we applied our method to the specification of a computer game. Results indicate that test machines can incrementally be extracted through a class hierarchy, and a parent test machine can be used to obtain its corresponding child test machines (by reusing test artifacts). Furthermore, by running extracted test cases on the implemented game, we discovered some real bugs.

Conclusion: The proposed approach can incrementally extract tests and bring extendibility and reusability as two main advantages of the object- oriented paradigm to the test domain.

1. Introduction

The nature of the object-oriented development process is iterative and incremental [1]. There are popular object-oriented analysis and design (OOAD) approaches for analyzing, designing and implementing a software system; through such approaches, iteration by iteration, the software artifacts will be refined continuously. Thus, as Bier says in [1], this iterative and incremental nature is well-suited to the test activity as we proceed in the development process. We should start testing as early as we can through the object-oriented development life cycle and refine the tests during the development process. Test cases can be extracted early in the process, even as requirements are being determined or changed; thus, faults can be detected early in the development process, saving time, money, and effort [1].

Moreover, object-oriented programming languages and concepts can lead to different types of errors raising the need to different approaches of testing [1]; even typical features of object-oriented languages (like message passing and encapsulation) are complex and therefore error-prone [1,2]. Concepts such as inheritance, polymorphism and dynamic binding can lead to data anomalies and failures which cannot be found by traditional testing techniques effectively [3,4]; to read more about these new types of data anomalies, refer to [5]. To discover such failures, we must extract tests which cover all different object-oriented testing levels, i.e., intra-method, inter-method, intra-class and inter-class levels.

Specification-based testing means extracting test cases from specifications rather than implementations. In specification-based testing, the test process can be started in an early stage of the

* Corresponding author..
E-mail address: h_haghighi@sbu.ac.ir (H. Haghighi).

http://dx.doi.org/10.1016/j.infsof.2016.05.005
0950-5849/

software development life cycle that allows more effective planning and utilization of resources [6]. Moreover, formal specifications can help in automating the test activity [6]. Although fully formal development methods are generally less popular and more used for the development of critical systems, even tiny formal specifications of important system modules can lead to great achievements. One of these achievements is the automated extraction of test cases in the early stages of the system development which reduces the cost of testing significantly [6]. Test oracles can be generated from formal specifications automatically [7]. Formal specification based oracles address all the oracle automation challenges [7] and are appropriate solutions to the oracle problem [3].

Most of the existing methods for extracting test cases based on formal, object-oriented specifications have been adapted from previous structural techniques and have not originally been presented for the object-oriented paradigm; therefore, they do not treat an object-oriented specification as an extendible and modifiable artifact through the object-oriented development process. Hence, there is no formal specification based test case generation method which can be effectively applicable through the incremental and iterative development process. More precisely, there is no method for deriving initial test artifacts based on formal specifications and then evolving these test artifacts based on the evolution of specifications.

The object-oriented paradigm relies heavily on the reusability concepts; examples are the inheritance mechanism in code and techniques for reusing design artifacts to save effort through the object-oriented development. But existing testing approaches do not use reusability opportunities in order to save cost and time through the testing phase. One of such opportunities is reusing parent class tests to test child classes through inheritance.

Although a tiny formal specification or formal model of an object-oriented software can support the testing approach to uncover some inter-class failures, most existing approaches do not use this opportunity and do not address data anomalies and failures which occur due to special object-oriented concepts (such as polymorphism and dynamic binding), and thus, cannot effectively be found by traditional testing techniques.

The main limitations of existing approaches can be summarized as the following list:

1. There is no method which can be applicable and adaptable through the incremental and iterative object-oriented development process. Such a method should be flexible against changes in software specifications and software models, and should provide a mechanism to evolve test artifacts when related artifacts evolve.
2. The lack of a comprehensive solution that covers all different object-oriented testing levels including intra-method, inter-method, intra-class and inter-class levels.
3. The lack of a solution which supports opportunities to reuse test artifacts through the object-oriented development process. A well-known opportunity is reusing test cases from ancestor classes to test descendant classes. To the best of our knowledge, only one work [11] concerns this opportunity, but this work reuses just some basic test information form parent classes only in special circumstances. With this inherited test information, we still need much effort to obtain actual tests which means this work does not have a tangible impact on the testing cost reduction.
4. The lack of effective test cases dealing with issues of inheritance, polymorphism and dynamic binding which have considerable potential for data anomalies and failures; although there are a few methods (to the best of our knowledge, only three works [8,9] and [10]) that deal with polymorphism and dynamic binding issues, they only address some basic aspects of the polymorphism issues.
5. In addition, few existing methods have been described in an algorithmic manner and with sufficient details needed for automation, while one of the major advantages that come from specification-based testing is the potential for automation.

Focusing on issues 1 and 2, this paper presents a new incremental method to extract test cases for all different object-oriented testing levels based on formal, object-oriented specifications written in Object-Z. Of course, the proposed method does not rely substantially on a particular formal, object-oriented specification language, or in other words, one can apply it to other formal methods (like JML specification in java or even formal graphical models like UML/OCL or UML-B) with minor modifications. For each operation of a class, we make a kind of finite state machine (FSM) that is called “test machine” and addresses the intra-method test level; then, with incremental combination of these test machines, we make a test machine for a class that covers inter-method and intra-class test levels. For extracting a test machine for a class, we introduce three algorithms: one for extracting test machines for individual operations. The other two algorithms work together to combine the resulting test machines. With this algorithmic presentation, we address the issue number 5 from the above issue list.

Addressing the issue number 1, our incremental approach brings extendibility [12] from the object-oriented domain to the test domain. A test machines obtained by our incremental approach can be easily adapted when the class specification is modified (for example, in case of operation modification, operation deletion, operation addition, class state variables addition and so on). This incremental nature can make our approach appropriate for using through incremental and iterative object-oriented development methodologies. This also makes it possible to reuse an extracted test-machine through the object-oriented inheritance mechanism. We mention how to use the proposed approach for easily extracting a test machine for a child class directly from the test machine of its parent class; with this, we address the issue number 3 and bring the inheritance concept from the object-oriented paradigm to the test domain.

Regarding the issue number 4, although a test machine for a class covers intra-class tests, we state how to use the proposed approach to extract test cases for some inter-class tests that deal with polymorphism and dynamic binding issues. Due to the space limitation, in this paper, we tackle issues 1, 2, and 5, and concentrate on the primary aspects of our approach and only present an overall view of its application regarding inheritance and polymorphism. The details of the application of our approach to inheritance, polymorphism, and dynamic binding issues with sufficient examples will be presented in the future work.

The paper is organized as follows. Section 2 reviews the related works. Section 3 provides some background notions. Section 4 presents our incremental method for extracting test cases. Section 5 briefly describes how to use the proposed method in the presence of inheritance (in order to reuse existing tests) and how to deal with polymorphism and dynamic binding issues. Section 6 is devoted to the evaluation of the proposed method. And finally, Section 7 concludes the paper.

2. Related work

Dick and Faivre [14] proposed an approach to convert VDM expressions into a disjunctive normal form (DNF) by which a domain space partitioning to derive test requirements can be done. One of the advantages of this approach is providing a potential for automation. Although the approach is for non-object-oriented