The Journal of Systems and Software 000 (2017) 1–14

Contents lists available at ScienceDirect

The Journal of Systems and Software

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

A multi-level feedback approach for the class integration and test order problem

Miao Zhang[a], Shujuan Jiang[a,*], Yanmei Zhang[a,b], Xingya Wang[a], Qiao Yu[a]

[a] School of Computer Science and Technology, China University of Mining and Technology, Xuzhou, China
[b] Guangxi Key Laboratory of Trusted Software, Guilin University of Electronic Technology, Guilin, China

ARTICLE INFO

Article history:
Received 27 February 2017
Revised 1 August 2017
Accepted 12 August 2017
Available online xxx

Keywords:
Test order
Test cost
Stub minimization
Feedback

ABSTRACT

Class integration and test order (CITO) problem is to devise an optimal inter-class order which can minimize stubbing efforts. The existing approach for this problem, whether it is graph-based or search-based, usually wastes a significant amount of time and efforts in finding test orders, and sometimes may devise sub-optimal solutions. To overcome this limitation, we introduce a multi-level feedback approach to better solve the CITO problem. In this method, we use a multi-level feedback strategy to calculate test profit for each class, and according to test profit, propose a reward and punishment mechanism to assess the performance of class and set its test priority. Instead of breaking cycles or searching for optimum in the previous methods, our method integrates classes by their test priority, therefore significantly reduces the running time. The experiments are conducted on five benchmark programs and eight industrial programs, and the obtained results are compared with graph-based and search-based approaches. The results indicate that our approach can minimize the stubbing cost efficiently for most programs of all typical approaches compared in this work.

1. Introduction

In the integration test of object-oriented (OO) software, the order to integrate and test classes is closely related to the stubbing efforts. Stubs need to be constructed for class A to emulate the behavior of another unavailable class B that is depended on by class A, during the test of A. In particular, when class A and class B are interdependent, constructing stubs for which class depends on the test order between class A and class B. Due to the complicated inter-class dependencies like this, the creation of stubs is inevitable, which is also expensive and error-prone (Labiche et al., 2000). Therefore, to construct fewer stubs at lower stubbing cost, classes should be integrated and tested by an optimal order.

In the literature, different kinds of approaches were proposed to devise optimal integration test orders. Graph-based algorithms (Kung et al., 1995; Tai and Daniels, 1997; Le Traon et al., 2000; Briand et al., 2001) generally construct an Object Relation Diagram (ORD) to identify Strongly Connected Components (SCCs), break cycles in non-trivial SCCs and test classes by reverse topological ordering of classes considering their dependencies. Graph-based algorithms are easy to understand, but they present two limitations: (1) although they can reduce the number of stubs, they tend to construct more complex stubs in many cases (Briand et al., 2002a); (2) recursively identifying SCCs and breaking cycles add more time and efforts, especially when the systems contain hundreds of thousands, or even millions of cycles. For that reason, search-based algorithms were proposed in Briand et al. (2002b), Borner and Paech (2009), Wang et al. (2010) and Steindl and Mottok (2012). This kind of approaches randomly generates initial individuals (class integration test orders), produces new offspring by using some evolution operators, and selects more adequate solution (the integration test order with minimum stubbing complexity) by calculating and comparing their fitness. Search-based algorithms are flexible because they no longer break cycles, but the results are not deterministic and sometimes they waste time on searching when trapped into local optimum particularly for large-scale programs.

As the aforementioned approaches take too much time to generate integration test orders, we introduce an efficient method named MLFCITO (Multi-Level Feedback Approach for the Class Integration and Test Order). This method can reduce the stubbing cost at less time cost, and provide a time-saving test order for the integration test. Considering the time cost is mainly caused by breaking cycles in graph-based algorithms and searching for optimum in search-based algorithms, MLFCITO drops these two measures and leverages an incremental strategy that classes are integrated according to their test priority. MLFCITO initially sets test

* Corresponding author.
E-mail address: shjjiang@cumt.edu.cn (S. Jiang).

http://dx.doi.org/10.1016/j.jss.2017.08.026
0164-1212/© 2017 Elsevier Inc. All rights reserved.

priority for each class by reward and punishment mechanism, and dynamically adjusts priority for untested classes based on their test profit. This test profit is calculated according to the feedback information of inter-class dependencies between the untested and tested classes. To decrease the stubbing cost, test priority for each class is set and adjusted according to its stubbing complexity. To save time for the integration test, classes in the same test priority can be integrated and tested together, instead of stepwise. In fact, testers can sort these classes again to meet their practical demands, without increasing stubbing cost. e.g., they can integrate multiple classes at once; or they can test the high demand classes at first.

We perform experiments on five benchmark systems and eight large real systems with varying complexities to evaluate MLFCITO. The results are very encouraging as the new proposed approach can clearly reduce the stubbing cost and execution time when compared to other graph-based algorithms and search-based algorithms.

The contributions of this work are as follows:

- We propose a novel, multi-level feedback approach named MLFCITO for the CITO problem. To our best knowledge, the strategy is entirely different from the previous graph-based and search-based approaches.
- We evaluate the effectiveness and efficiency of MLFCITO on benchmark programs and industrial programs, and compare it with the previous studies on a new aspect of stubs distribution. The results have confirmed the better performance of the proposed method.
- We provide a more flexible solution for the integration test. Based on the test order generated by MLFCITO, testers can meet their practical demands at a lower stubbing cost, especially in some cases, they can perform a parallel integration test.

The paper is organized as follows. Section 2 presents a review of works related to the class integration and test order problem. Section 3 introduces MLFCITO. Section 4 represents the empirical evaluation conducted, its results and analysis. Finally, Section 5 concludes this work and discusses future works.

2. Related work

CITO problem can be defined as to devise an optimal order for integrating and testing the classes (Abdurazik and Offutt, 2006). Such order means that the number of created stubs and the stubbing complexity are lowered to the possible minimum. So works that address CITO problem generally attempt to minimize the number of stubs, or to reduce the stubbing complexity (Wang et al., 2011), which can be characterized by two different techniques: graph-based and search-based. In this section, an overview of these works is presented.

The graph-based approach proposed by Kung et al. (1995) was the first one to address this problem. They broke cycles by removing association edges in cycles, and then produced test order by topological sorting for the acyclic digraph. When there was more than one association edge that could be removed for cycle breaking, they randomly selected any one. However, removing different association edges would construct varying numbers of stubs, and sometimes randomly removing an association might lead to higher stubbing cost. To create fewer stubs, the association edges that were involved in more cycles would be removed. Therefore, researchers (Tai and Daniels, 1997; Le Traon et al., 2000; Briand et al., 2001) used weights to measure the relation between edges (or vertices) and number of cycles, and proposed diverse approaches to break cycles by removing candidate edges or vertices according to their weights.

Tai and Daniels (1997) first integrated classes according to their major-level numbers which were determined only based on strongly connected relations (such as, inheritances and aggregations). Then for the classes at the same major level, they iteratively removed the association edge with the highest weight until no cycles existed, and tested classes according to minor-level numbers which were determined based on weakly connected relations such as associations. Every association was assigned a weight based on related incoming and outgoing dependencies to approximately measure the number of cycles it involved in. Le Traon et al. (2000) identified frond edges (an edge starting from vertices to their ancestors) to assign weights for vertices, and removed all incoming edges of the vertex with the highest weight to break cycles. Compared with other approaches, this approach may remove strongly connected relations leading to the construction of complicated stubs. Briand et al. (2001) removed only association edge with the highest weight, and improved Tai and Daniels' approach in weight computation. They took the product of incoming dependencies and outgoing dependencies as edges' weights to measure the number of involved cycles, which was more precise than Tai and Daniels' weight computation.

The above three approaches are non-deterministic. For Le Traon's approach (Le Traon et al., 2000), vertices' weights depend on the number of related frond edges, and such edges are determined by the traversing order of vertices. Each time the results obtained would be uncertain since whether an edge is removed depended on how the ORD is traversed. Other two approaches perform a random selection when there are two or more candidate associations with the same weights, which results in non-deterministic test orders.

Hewett and Kijsanayothin (2009) proposed a deterministic approach which incrementally selected the appropriate classes for the integration test. Similar to ours, this approach requires neither the identification of SCCs nor the breaking of cycles. However, the approach of Hewett and Kijsanayothin (2009) has three main differences from ours. First, Hewett et al.'s method aims at reducing the number of stubs while our purpose is to minimize the stubbing complexity. Second, in Hewett et al.'s method, classes and dependencies are treated as vertices and edges in a Test Dependency Graph (TDG), CITO problem is simplified to the problem of sorting the vertices in TDG. Compared with us, they are less concerned with the information of systems, and neglect to differentiate types of dependencies. Third, in Hewett et al.'s method, the number of stubs required by class A is approximately measured by vertex A's outdegree. While we measure the stubbing complexity based on the attribute coupling and method coupling.

The objective of the above approaches is to minimize the number of test stubs to be constructed, but sometimes they may construct extra complicated stubs, because actually stubbing cost is also affected by other factors, such as number of attributes accessed, and number of methods invoked (Briand et al., 2002b). Hence, Briand et al. (2002a) measured stubbing complexity of dependencies by the numbers of attribute coupling and method coupling. Abdurazik and Offutt (2006) introduced nine kinds of couplings and considered more information in measuring the stubbing efforts, including the number of parameters, the number of return value types and the number of variables, which improved the accuracy but increased the complexity of the algorithm. Jiang et al. (2012) reassigned the weights for the four coupling information to reduce algorithm's complexity while guaranteeing the accuracy. Indeed, the coupling information introduced by these three approaches is the same, since attribute coupling proposed by Briand et al. includes the return value type coupling and parameter coupling. Considering that Briand's coupling measures are more popular, the approach proposed in this paper adopts Briand's strategy in measuring stubbing cost.