Methods and Metrics for Selective Regression Testing

Rami Bahsoon [+]* and Nashat Mansour*
[+]*Department of Computer Science, University College London, Gower Street, London, WC1E 6BT, UK. E-mail: r.bahsoon@cs.ucl.ac.uk*
**Computer Science Program, Lebanese American University, P.O. Box 13-5053, Beirut, Lebanon. E-mail: nmansour@lau.edu.lb*

Abstract

In corrective maintenance, selective regression testing includes test selection from previously run test suite and test coverage identification. We propose three reduction-based regression test selection methods and two coverage McCabe-based identification metrics. We empirically compare these methods with other three reduction and precision-oriented methods using 60 test-problems. The comparison shows that our proposed methods yield favourable results.

1. Introduction

Let $TS=\{t_1, t_2, \ldots, t_N\}$ be the set of N test cases used in the initial development of a program P. Selective regression testing addresses two major problems: test selection and coverage identification. Test selection problem requires a subset of test cases, R, be selected from TS for rerunning on the modified program P'. The objective is to provide confidence that no adverse effects have been caused by the modification. The coverage identification problem identifies portions of P' that require additional testing. This might require creating R', a set of new tests for P', and updating TS. This paper addresses both problems.

A number of selective regression testing methods have been proposed for guiding the test selection. In particular, Mansour and El-Fakih [4] have proposed using an optimization formulation of the selective retesting problem and a Simulated Annealing (SA) algorithm for minimizing the number of selected test cases. Harrold, Gupta, and Soffa [3] have suggested a Reduction methodology (RED) for managing a test suite, which can be used for reducing the number of selected test cases. Agrawal, Horgan, and Krauser [2] have proposed slicing algorithms (SLI) that select test cases whose output may be affected by the modification made to the program.

To address the test selection problem, we propose three reduction-based selective regression testing methods. These are Modification-Based Reduction version 1 (MBR1), Modification-Based Reduction version 2 (MBR2), and Precise Reduction (PR). MBR1 improves the RED algorithm by accounting for the location of the modification made and its effects. MBR2 improves MBR1 by selecting only test cases that execute the modification. PR uses slicing in a similar way to the SLI algorithm to determine the useful test cases and applies a reduction procedure to reduce the final number of selected retests. We use 60 test-problems to empirically evaluate MBR1, MBR2, and PR and compare them with SA, RED, and SLI.

We approach the coverage identification problem by suggesting two McCabe-based regression test coverage metrics. These are Reachability regression Test selection McCabe-based metric (RTM), and data-flow Slices regression Test McCabe-based metric (STM). RTM provides an upper-bound indication of the number of paths that must be tested. STM is a data-flow McCabe-based variable-dependent metric. It computes two bounds: an upper and a lower bound of the regression tests required for covering the affected definition-use pairs.

The rest of the paper is organized as follows. Section 2 describes the program modelling used. Section 3 presents the reduction-based methods. Section 4 presents the regression test coverage metrics. Section 5 gives the experimental results. Section 6 contains our conclusions.

2. Program modelling

A subject program is modelled by a control-flow graph G with n nodes. Each node represents a control statement or a contiguous sequence of assignment statements. The nodes stands for requirements $\{r_1, r_2, \ldots, r_n\}$ to regression test for. The graph edges represent control/data flow. $TS = \{t_1, t_2, \ldots, t_N\}$ can be expressed as a union of non-disjoint subsets $T_1 \cup T_2 \cup \ldots \cup T_n$, where $T_i=\{t_{i1}, t_{i2}, \ldots, t_{Ci}\}$ is a subset of TS used to cover requirement r_i. the cardinality of T_i is C_i. To perform slicing and data-flow based analysis, we extend the model to keep variables definitions and uses information in relevant segments/nodes.

3. Reduction-based methods

We propose three reduction-based methods that improve RED algorithm by accounting for the modification made and its effects. These are MBR1, MBR2, and PR. We only present the motivations of these methods due to space limitation.

MBR1 reduces the number of selected regression tests by eliminating tests that cover requirements impacted by

0-7695-1165-1/01 $10.00 © 2001 IEEE

the change and those that are redundant [3]. Our approach to determine the requirements potentially impacted by the change uses reachability information derived from the subject program's control-flow graph. Reachability information characterizes requirements that reach or are reachable from the modified one as potentially affected. MBR1 only tests for such. It selects a subset R of test cases from TS. R must contain at least one test case from each T_i associated with the affected r_i to secure an adequate coverage of the changed and potentially affected requirements.

MBR2 improves MBR1 by eliminating tests that cover a requirement characterized as potentially affected and do not reveal any modification. MBR2 uses these observations, derived from MBR1, to eliminate such tests: (i) not every test $t_k \in T_i$ exercising a requirement, which reaches or is reachable from the modified requirement, does necessarily cover the modification; (ii) if a requirement, say the modified one, is not executed by a test case, it can not affect the program output for that test; (iii) a requirement that reaches or is reachable from the modified requirement/segment does not necessarily affect the program output of the modification-related part of the subject program. For example, a control node initially characterized as potentially impacted by the change, might evaluate to unaffected requirements. Tests covering these paths might not reach the modification. MBR2 eliminates such cases.

PR uses modification and computed relevant slices [2] information. PR is motivated by the following observations: (i) not all requirements in the program are executed under a test case; (ii) if a requirement is not executed under a test case, it can not affect the program output for that test case; and (iii) even if a requirement is executed under a test case, it does not always affect the program output for that test case. These observations are used to orient the reduction process to only select tests with relevant slices that contain modified requirement(s) and affect the output for the considered modification(s). Such orientation ensures that only modification-revealing tests are the selected, all non-modification-revealing tests are omitted. Thus, the method is said to be precise [1]. PR also eliminates all redundant tests.

4. McCabe-based identification metrics

We suggest two McCabe-based [7] test selection coverage identification metrics. The metrics quantify the retesting effort; monitor retesting coverage adequacy; suggest bounds on regression tests; assist in generating tests to rerun for testing the change; and identify where additional tests may be required. The Cyclomatic complexity [7] is given by v (G) for a control-graph G.

The Reachability regression Test selection McCabe-based metric (RTM) provide an upper bound of the number of selected regression tests that guarantee the coverage of requirements potentially affected by the modification at least once. Given that segment/node l has been modified in G, The RTM upper bound, denoted by *RU (l)*, computes as follows. First, remove all control segments that do not reach and are not reachable from l. This yields a reduced flow graph G_1 with the same entry and exit nodes as those of G. *RU (l)*=v (G_1) provides an upper bound of the number of linearly independent paths that must be retested and hence the number of tests that must be rerun to guarantee coverage of all segments potentially affected by the change in l at least once.

In Data-flow Slices Regression Test McCabe-Based Metric (STM), we extend McCabe complexity to deal with data/variable modifications. STM has two bounds: upper and lower numbers of regression tests to cover the affected definition-use pairs due to data modification.

STM upper bound, denoted *a SU (x/l)*, assumes that the set of affected definitions-use pairs is identified using backward/forward algorithms of [6] due to the modification of variable x in segment l of G. Removing from G all paths with control segments not leading to the identified affected def-use pairs, we obtain a reduced graph G_1. *SU (x/l)*=v (G_1) gives the number of all possible independent paths from the entry node to the exit node of the subject program that cover the affected def-use pairs. Hence, the number of tests to rerun upon the modification of variable x in segment l.

To find a STM lower bound, we adjust the entry and exit points of the reduced program's graph, without losing affected def-use testing coverage. We suggest backward and forward adjustment of entries.

When a definition variable x is modified in segment/node l, we use forward adjustment: we set the node containing the modified variable as an adjusted virtual entry point. A forward-walk algorithm [6] is then performed to locate the uses of the variables x along all paths. To find an adjusted virtual exit node, we examine the number of returned affected def-use pairs of the modified variable x edited at l. If the returned number by the walk is one, then we set the node containing the only use as an adjusted virtual exit point. Else (if it exceeds one), we find a node to which the obtained uses converge. To determine such a node, the obtained uses are marked. Depth-first search is then performed from the marked uses to the first node at which searched paths converge. This node is marked as the adjusted virtual exit node.

When a use is modified, we perform backward adjustment. Backward adjustment follows the same concept of that of forward. But, it sets the node containing the modified use variable as an adjusted virtual exit node. It uses backward-walk of [6] to locate the definitions of the variables x along all paths and reverse-depth-first instead to converge to a virtual entry node.

STM lower bound computes *as SL (x/l)*=v (G_1), where G_1 is a reduced graph with virtual entry and exit nodes. This gives the number of all possible subpaths that must be tested to ensure the coverage of affected def-use pairs (from the virtual entry node to the virtual exit node). The STM lower bound suggests rerunning *SL (x/l)* as the minimum number of tests.

RTM and STM bounds can be employed to guide the test selection process for coverage-based regression testing methods. For example, data-flow methods of [6] tends to include all redundant tests in the regression suite

satisfying the affected def-use pairs and, in some cases, leads to a high number of selected tests. This fact has been experimentally observed [5]. To select a reduced regression test suite with redundant tests eliminated, we recommend using STM bounds to guide the test selection process.

5. Empirical results for test selection methods

We have generated 60 test problems from 17 program modules to empirically compare MBR1, MBR2 and PR with three reduction and precision-oriented methods. These are SA, RED, and SLI. We base our comparison on the following four quantitative criteria. (i) Percentage number of selected tests, *#R*; (ii) Execution time of the code implementing the method, t_{exec}; (iii) Precision to measure the method's ability to omit non-modification-revealing tests [1]; and (iv) Inclusiveness to measure the extent a method selects modification-revealing tests [1]. In Tables 1 and 2, we present aggregate results for the 60 test problems.

Table 1 shows that the PR algorithm yields the least *#R* 72% of the time. This is because PR omits all non modification-revealing and redundant tests. PR is the fastest (t_{exec}). MBR1 shows slight improvement in #R over RED. Because most of the modules used exhibit strong inter-segment dependency, MBR1 tends to test for almost all segments in a similar way to RED. The results show that MBR1's *#R* varies with the location of the modified segment and its degree of reachability to other segments. MBR2 selects fewer tests when compared to MBR1 due to its ability to omit tests that do not reach the modification.

Table 1. Aggregate results for #R and t_{exec}

% Times	SA	RED	SLI	PR	MBR1	MBR2
Least #R	43%	10%	37%	72%	13%	37%
Worst #R	10%	55%	52%	0%	47%	22%
Fastest t_{exec}	0%	73%	0%	95%	87%	75%
Slowest t_{exec}	15%	0%	83%	0%	0%	0%

Table 2. Inclusiveness and precision aggregate results

% Times	SA	RED	SLI	PR	MBR1	MBR2
Highest Precision	13%	13%	100%	100%	18%	40%
Lowest Precision	30%	78%	0%	0%	60%	5%
Highest Inclusiveness	12%	15%	83%	12%	12%	21%
Lowest Inclusiveness	68%	54%	18%	37%	46%	21%

Table 2 shows that PR (like SLI) is fully precise. This is due to its ability to omit all non-modification revealing test cases and select only ones that are modification revealing. The inclusiveness of PR depends on the selected test cases with relevant slice containing a modified requirement and is generally low; if there are several redundant test cases with relevant slices traversing a modified segment, PR attempts to select only one of them. MBR2 omits tests that fail to execute the modification and shows improved precision over MBR1 and RED. When testing for programs with segments exhibiting low reachability to the modified one, MBR1 tends to give higher precision values than that reported by RED. The inclusiveness of SA, RED, MBR1, and MBR2 is low, since: (i) RED does not explicitly target modification revealing test cases due to its independence of the location of the modified segment, and (ii) if several modification revealing test cases traverse a particular segment, RED, MBR1, MBR2 and SA attempt to select only one of them.

6. Conclusions

We have proposed three reduction-based regression test selection methods (MBR1, MBR2, and PR) and two McCabe-based metrics (RTM and STM) for regression test coverage identification. We have empirically compared MBR1, MBR2, and PR to three reduction methods: SA, RED, and SLI. The results show that three methods offer reduction in the test suite. In contrast with RED, they are dependent on the modification made to the subject program. PR selects the least number of tests for most the test problems with full precision. MBR2 selects fewer tests than RED and MBR1 most of the time. PR selects fewer tests than SLI while maintaining full precision like SLI.

The RTM metric is an upper bound metric. It provides an indication of the number of potentially affected paths that must be retested. The STM metric yields an upper and a lower bound on the number of tests to rerun for testing the affected definition-use pairs.

References

[1] G. Rothermel, and M.J. Harrold, "Analyzing regression test selection techniques", *IEEE Trans. Software Engineering*, 22(8), August 1996, pp. 529-551.

[2] H. Agrawal, J.R. Horgan, and E.W., Krauser, "Incremental regression testing", In: *Proc. Conference on Software Maintenance*, 1993, pp. 348-357.

[3] M.J. Harrold, R. Gupta, and M.L. Soffa, "A methodology for controlling the size of a test suite", *ACM Transaction on Software Engineering and Methodology*, July 1993, pp. 270-285.

[4] N. Mansour, and K. El-Fakih, "Simulated annealing and genetic algorithms for optimal regression testing", *Journal of Software Maintenance*, 1999, Vol. 11, pp. 19-34.

[5] N. Mansour, R. Bahsoon, and G. Baradhi, "Empirical comparison of regression test selection algorithms". Accepted, to appear in: *Journal of Systems and Software*, 2001.

[6] R. Gupta, M.J. Harrold, and M.L. Soffa, "An approach to regression testing using slicing". In: *Proc. Conference on Software Maintenance*, 1992, pp. 299-308.

[7] T. McCabe, "A complexity measure", *IEEE Trans. On Software Engineering*, 1976, 2(3), pp. 308-319.