A Framework to Support Research in and Encourage Industrial Adoption of Regression Testing Techniques

Jonathan Miller Kauffman
Department of Computer Science
Allegheny College
kauffmj@allegheny.edu

Gregory M. Kapfhammer
Department of Computer Science
Allegheny College
gkapfham@allegheny.edu

***Abstract*—When software developers make changes to a program, it is possible that they will introduce faults into previously working parts of the code. As software grows, a regression test suite is run to ensure that the old functionality still works as expected. Yet, as the number of test cases increases, it becomes more expensive to execute the test suite. Reduction and prioritization techniques enable developers to manage large and unwieldy test suites. However, practitioners and researchers do not always use and study these methods due, in part, to a lack of availability. In response to this issue, this paper describes an already released open-source framework that supports both research and practice in regression testing. The sharing of this framework will enable the replication of empirical studies in regression testing and encourage faster industrial adoption of these useful, yet rarely used, techniques.**

***Keywords*-open-source framework; regression testing;**

I. Introduction

Software developers modify a program P to fix existing defects or add new features. The goal of regression testing is to ensure that the changes to P do not introduce faults into the old functionality by running a test suite $T = \langle t_1, t_2, \ldots, t_n \rangle$ [1] [2]. Each test case $t_i \in T$ exercises some part of P to ensure that the program still works as expected.

As the number of test cases in the test suite grows, re-executing all of the tests becomes increasingly expensive. There are two noteworthy techniques for managing regression test suites. The first is test suite reduction, which removes redundant test cases from the test suite in order to decrease execution time [1]. The second is test suite prioritization, which reorders test cases so that the ones which are run first are those that best fulfill some objective [2].

Despite the benefits of these regression testing methods, their lack of availability often limits their use in practice and causes research to stagnate [3]. Moreover, some researchers and practitioners are unwilling to use these algorithms unless they are freely available, fully documented, properly supported, and/or useful with minimal configuration. In order to address these issues, we have developed a framework consisting of two free and open-source (FOSS) tools: Proteja (*http://proteja.googlecode.com*) and Modificare (*http://modificare.googlecode.com*). Before we released this framework, there was no way to use these algorithms without independently implementing them from their descriptions in published research articles. This paper describes these two tools and explains how they encourage both research and practice in regression testing.

II. Open-Source Testing Tools

A. Proteja

Proteja ("protect" in Romanian) is a tool written in Java for executing and performing coverage monitoring on JUnit test suites. It supports three types of coverage criteria: statement, method, and class coverage. As illustrated in Figure 1, Proteja accepts a program and test suite as input, so that, after a negligible amount of configuration, a tester can use it to execute the tests and collect coverage and test case timing information. To acquire the per-test coverage reports, Proteja extends Cobertura [4], a widely-used and well-supported coverage monitor that alone cannot produce the per-test information needed by most regression testing methods.

In addition to executing entire test suites, Proteja can also be configured to run a subset of the test suite or to execute the test cases in a different ordering. The tool accepts a modified test suite $T^{'}$ that indicates which test cases are to be run and in what order they should be executed, as shown in Figure 1. Proteja can then execute the test suite according to the reduction or prioritization represented by $T^{'}$.

B. Modificare

Modificare ("modify" in Italian), a tool written in the R language for statistical computing [5], performs test suite reduction and prioritization and supports experimentation in regression testing. This tool provides implementations of six reduction and prioritization algorithms: random, adaptive random, greedy, hill climbing, simulated annealing, and genetic. As illustrated in Figure 1, Modificare accepts coverage and test case timing information in a language-independent format and performs either test suite reduction or prioritization, ultimately producing both a modified test suite $T^{'}$ and a fitness score indicating the quality of the reduction or ordering. If modifying the test suite improved its quality (e.g., a large percentage of seeded faults are

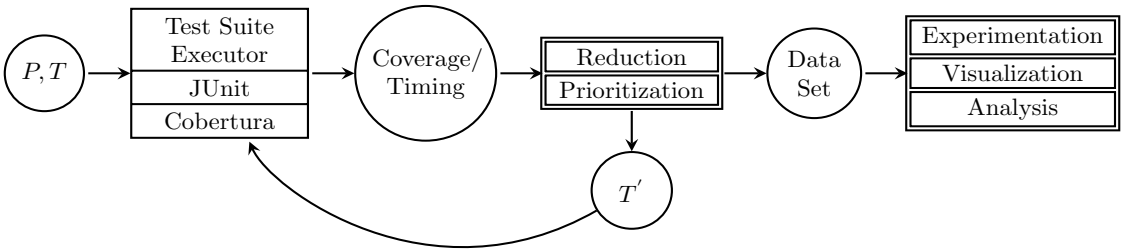

Figure 1. Open-source framework containing the Proteja (single-line box) and Modificare (double-line boxes) tools.

detected quickly), then $T^{'}$ can be provided as input to Proteja and used during subsequent rounds of test suite execution.

In addition, Modificare can run the reduction and prioritization techniques for a number of trials and in a variety of configurations. Since running such experiments can be time consuming[1], the tool also provides facilities for distributing the execution of experiments over a cluster of computers. Following the standards in [6], the data set produced by performing reduction or prioritization can be both visualized and statistically analyzed in order to highlight the most significant trends in the data. For example, the greedy algorithm produces slightly better results for certain programs, yet the hill climbing algorithm executes over four times faster than greedy. This suggests that a small decrease in algorithm effectiveness may result in a large increase in efficiency. We also found that randomly prioritizing the test suite for a Java application that solves sudoku puzzles caused the average percentage of faults detected to increase from 39% to 72%. This indicates that a simple prioritization strategy may significantly decrease the time needed to reveal a fault.

III. Supporting Research and Practice

A. Research

Proteja supports research in regression testing because it allows developers to easily and efficiently collect coverage information for a wide variety of applications, thus improving the external validity of empirical results [3]. While we chose coverage because it is impossible to reveal a fault unless it is first executed by the tests [7], other objectives, such as executing the transitions in a model, may be used as long as tools for acquiring per-test information are available.

Modificare supports research in regression testing because it allows data to be collected for algorithms run both in a wide variety of configurations and for many trials. In addition, support for the distributed execution of experiments will encourage researchers to perform studies that are greater in size and more thorough, allowing for more comprehensive comparisons to be made between reduction and prioritization algorithms. Also, the features of visualization and statistical analysis will aid researchers in identifying the most interesting and statistically significant trends in their data [3].

[1]For example, running the adaptive random algorithm in six configurations took 36.3 hours when experiment execution was distributed over 19 Ubuntu Linux workstations with a 3.06 GHz Intel Core 2 Duo processor.

B. Practice

Proteja supports practice in regression testing in the same way that it supports research — by allowing practitioners to efficiently collect coverage reports for Java programs, thus encouraging them to use regression testing techniques.

Modificare supports practice in regression testing because, after performing test suite reduction or prioritization, it returns a modified test suite that can be input to Proteja for further rounds of test suite execution and coverage monitoring.

Both tools are designed to be easy to use and amenable to configuration and extension and they currently integrate with widely-used open-source tools such as JUnit and Cobertura.

IV. Conclusion and Future Work

In order to encourage researchers and practitioners to use regression testing methods, we have released a framework consisting of two open-source tools: Proteja and Modificare. In future work, we will continue to refine and extend these tools. As an example, we plan to enhance the interface for integrating coverage monitors, thus allowing Proteja to support a variety of coverage criteria, such as definition-use and path coverage [8]. For Modificare, we plan to automate the process of experimentation, visualization, and statistical analysis and to enable testers to define and use their own fitness scores for guiding and evaluating regression testing.

References

[1] M. J. Harrold, R. Gupta, and M. L. Soffa, "A methodology for controlling the size of a test suite," *ACM Trans. on Softw. Eng. and Method.*, vol. 2, no. 3, 1993.

[2] S. Elbaum, A. G. Malishevsky, and G. Rothermel, "Test case prioritization: A family of empirical studies," *IEEE Trans. Softw. Eng.*, vol. 28, no. 2, 2002.

[3] G. M. Kapfhammer, "Empirically evaluating regression testing techniques: Challenges, solutions, and a potential way forward," in *Proc. of the 1st Intl. Wkshp. on Regr. Test.*, 2011.

[4] Cobertura, "Calculates the percentage of code accessed by tests." *http://cobertura.sourceforge.net/.*

[5] R Core Development Team, "R: A language and environment for statistical computing," *http://www.R-project.org*.

[6] A. Arcuri and L. Briand, "A practical guide for using statistical tests to assess randomized algorithms in software engineering," in *Proc. of the 33rd Intl. Conf. on Softw. Eng.*, 2011.

[7] J. M. Voas, "PIE: A dynamic failure-based technique," *IEEE Trans. Softw. Eng.*, vol. 18, no. 8, 1992.

[8] H. Zhu, P. A. V. Hall, and J. H. R. May, "Software unit test coverage and adequacy," *ACM Comp. Sur.*, vol. 29, no. 4, 1997.