

# A Risk-based Regression Test Selection Strategy

Yanping Chen, Robert L. Probert, *University of Ottawa, Ontario, Canada*

**Abstract**— Risk is anything that threatens the successful achievement of a project’s goals. The fundamental principle of *risk-based testing* is to do more thorough testing to those parts of the software system that present the highest risk. In this fast abstract, we introduce risk-based testing and discuss applying risk analysis to select test cases for regression testing which is essential to ensure software quality. We provide a method of risk-based test case selection. This approach is a specification-based method. Therefore, it does not have scalability problems as does code-based techniques. It is also easy to implement with test tools, thus, making the regression test process more automated.

**Index Terms**—Risk analysis, Software testing, regression analysis, requirement traceability.

## I. INTRODUCTION

REGRESSION testing is essential for ensuring software quality. It is the process of validating modified software to provide confidence that the changed parts of the software behave as intended and that the unchanged parts of the software have not been adversely affected by the modification [1]. Much attention has been given to this topic in recent years. Although existing research has addressed and solved some problems, most regression testing is code-based. Code-based regression test selection is good for unit testing, but it has a scalability problem. As the size of the system under test grows, it becomes harder to manage test information and to create corresponding traceability matrices.

Thus, although many researchers are investigating regression testing techniques, very few research projects involve regression test selection techniques that are specification-based [2, 3]. In practice, some companies, such as IBM, have developed specification-based regression test strategies.

In this note, we discuss a specification-based method for regression test selection. We use risk analysis to guide test case selection, and measure the quality of the regression test suite.

## II. BACKGROUND

Risk is a threat to the successful achievement of a project’s goals. The tester’s job is to reveal high-risk

problems in the product. Traditional testers have always used risk-based testing, but in an ad hoc fashion based on personal judgment [8]. Using risk metrics to quantitatively measure the quality of a test suite seems more reasonable and is our approach.

Some researchers, especially those from industry, are very interested in the idea of risk-based testing. Pfleeger [5] discusses risk management. Amland [6] presents fundamentals and metrics of risk-based testing. Bach works on methodologies for risk analysis [7].

## III. TEST SELECTION STRATEGY BASED ON RISK

Major goals of *regression testing* are to assure system stability and reliability. Our risk-based approach focuses on test cases which test the risky areas. Hence, it can help us **achieve adequate confidence in software quality**. Our test selection method consists of two parts: test case selection and end-to-end test scenario selection.

### A. A Sample Risk Model

In [6] Amland presented a simple risk model with only two elements in *Risk Exposure*. We use this model in our research. It takes into account both:

- 1) The probability of a fault being present.
- 2) The *cost* (consequence or impact) of a fault in the corresponding function if it occurs during normal operation.

The mathematical formula to calculate *Risk Exposure* is  $RE(f) = P(f) \times C(f)$ , where  $RE(f)$  is the *Risk Exposure* of function  $f$ ,  $P(f)$  is the *probability* of a fault occurring in function  $f$  and  $C(f)$  is the *cost* if a fault is executed in function  $f$  in operational mode.

### B. Risk-based Test Case Selection

Using the risk model presented in the previous section, our Risk-based Test Selection approach has four main steps.

**Step 1:** Estimate the cost for each test case:

*Cost* is categorized on a one to five scale, where one is low and five is high. Two kinds of costs will be taken into consideration:

- 1) The consequences of a fault as seen by the customer, for example, losing market share because of faults,
- 2) The consequences of a fault as seen by the vendor, for example, high software maintenance costs because of faults.

**Step 2:** Derive *severity probability* for each test case:

For each test case, we compute *severity probability* depending on the number of defects and the severity of defects. *Severity probability* falls into a zero to five scale, where zero is low and five is high.

**Step 3:** Calculate *Risk Exposure* for each test case:

Risk Exposure is the multiplication of *Cost* and *severity probability*. To enhance or focus the results, we can also add weights to test cases that we need to give preference to.

**Step 4:** Select test cases that have the highest values of *Risk Exposure* as *Safety Tests* (Safety Tests check that serious failures do not occur).

### C. Risk-based Test Scenario Selection

Since end-to-end scenarios involve many components of the system working together, they are highly effective at finding regression faults. Our selection strategy obeys two rules:

**R1:** Select scenarios to cover the most critical test cases.

**R2:** Ensure scenarios cover as many test cases as possible.

Based on a traceability matrix showing the relation between end-to-end test scenarios and test cases, our risk-based test scenario selection approach also has four main steps.

**Step 1:** Calculate Risk Exposure REs for each scenario:

Using the traceability matrix, we can simply calculate the *Risk Exposure* for each scenario by summing up the *Risk Exposure* for all test cases that this scenario covers.

**Step 2:** Select the scenario with highest REs

**Step 3:** Update the traceability matrix (remove selected scenarios and covered test cases, and *re-calculate REs*):

When running the chosen scenario, all test cases covered by the scenario will be executed. According to our selection rules, these test cases should not affect our selection any more. We should focus on those test cases that have not yet been executed. In our approach, we cross out the column for the chosen scenario, and rows for all the test cases covered by the scenario. All of this is easily automated.

**Step 4:** Repeat Steps 2 and 3 as desired.

## IV. CONCLUSION

In this note we have presented a *specification-based (black box)* regression test selection technique. This technique is *customer-oriented* and also *risk-based*. It is also easy to implement with test tools, thus, making the regression test process more automated. Our industrial case studies indicated that the technique is effective in finding defects. Details about our approach and research results can be found in [9] and [10].

Our future work includes using more experimentation and empirical case studies to enhance the effectiveness of our technique, developing new metrics (for example, specification coverage) to help decide when to stop regression testing, and implementing our approach in a production test process and guide test selection. We will also use commercial statistical analysis tools to measure the cost-effectiveness in practice.

## REFERENCES

- [1] Mary Jean Harrold, James A. Jones, Tongyu Li, and Donglin Liang, "Regression Test Selection for Java Software", *Proc. of the ACM Conf. on OO Programming, Systems, Languages, and Applications (OOPSLA '01)*, 2001, pp. 312 – 326.
- [2] H.K.N. Leung and L.J. White, "A study of integration testing and software regression at the integration level", *Proc. of the Conf. on Software Maintenance*, November 1990, pp. 290 – 300.
- [3] Anneliese von Mayrhauser, Richard Mraz, Jeff Walls, and Pete Ocken, "Domain Based Testing", *Proc. of the Conf. on Software Maintenance*, September 1994, pp.484 - 491.
- [4] James Rumbaugh, Ivar Jacobson, and Grady Booch, *The Unified Modeling Language Reference Manual*, Addison Wesley Inc., 1999.
- [5] Shari Lawrence Pfleeger, "Risky Business: What We Have Yet to Learn about Risk Management", *Journal of Systems and Software*, Vol. 53, 2000, pp. 265-273.
- [6] Stale Amland, "Risk Based Testing and Metrics: Risk analysis fundamentals and metrics for software testing including a financial application case study", *Journal of Systems and Software*, Vol. 53, 2000, pp. 287 - 295.
- [7] James Bach, "Heuristic Risk-Based Testing", *Software Testing and Quality Engineering Magazine*, November 1999, pp. 96 - 98.
- [8] John D. McGregor, and David A. Sykes, *A Practical Guide to Testing Object-Oriented Software*, Addison Wesley Inc., 2001.
- [9] Yanping Chen, Robert L. Probert, D. Paul Sims, "Specification-based Regression Test Selection with Risk Analysis", in *the Proceeding of CASCON'02*, September 2002, pp..
- [10] Yanping Chen, *Master's thesis: Specification-based Regression Test Selection with Risk Analysis*, University of Ottawa, December 2002.

**Yanping Chen** received her Master's degree in computer science from University of Ottawa, Canada, in 2003, and her B.A.Sc. in computer science from the East China Normal University, China, in 1995. She is currently a PH.D student (Computer Science) in SITE, the School of Information Technology and Engineering at the University of Ottawa. She is working on software testing under the supervision of Dr. Robert Probert.

In 2002, Yanping was awarded a CITO fellowship to work on Regression Testing in IBM Toronto Lab. This year, she holds a doctoral fellowship from IBM. Since January 2002, she has worked as a software test process improvement advisor in the Electronic Commerce Department, IBM. Her interests include formal verification and validation, regression testing, automatic test case generation, and TTCN.

**Robert L. Probert** (Ph.D., Computer Science, University of Waterloo, 1973) is Full Professor and was the founding Director of the School of Information Technology and Engineering (SITE) and Coordinator of the Nortel Networks ASERT Lab at the University of Ottawa. He has been a Visiting Scientist with IBM CAS since 1998, and is now a Faculty Fellow in E-Commerce Testing. His research interests and publications are primarily in testing protocols and communications software quality engineering.