On Increasing System Test Effectiveness through a Test Case Prioritization Model Using Static Metrics and System Failure Data

Laurie Williams[1], Will Snipes[2], Andrew Meneely[1]
[1] *Department of Computer Science, North Carolina State University, Raleigh, NC, USA*
{lawilli3, apmeneel}@ncsu.edu
[2] *Nortel Networks, Research Triangle Park, NC, USA*
wbsnipes@nortel.com@nortel.com

Abstract

System testing is the last phase before the product is delivered for customer use and thus represents the last opportunity for verifying that the system functions correctly and as desired by customers. System test is time consuming in that it involves configuring and testing multiple complete, integrated systems (including hardware, operating system, and cooperating and co-existing applications) that are representative of a subset of customer environments. As a result, prioritizing the execution order of system test cases to maximize system test effectiveness would be beneficial. We are developing a statistical test case prioritization model that uses static metrics and system failure data with the goal of improving system test effectiveness.

1. Introduction

Software testing is a strenuous and expensive process [2, 3]. Research has shown that at least 50% of the total software cost is comprised of testing activities [6, 14]. System testing is testing conducted on a complete, integrated system to evaluate the system compliance with its specified requirements [10]. Configuring and testing multiple complete, integrated systems (including hardware, operating system, and cooperating and co-existing applications) that are representative of a subset of customer environments is time intensive. System testing is the last phase before the product is delivered for customer use and thus represents the last opportunity for verifying that the system functions correctly and as desired by customers [4]. Additionally, companies are often faced with lack of time and resources, which can limit their ability to effectively complete testing efforts. Prioritization of the execution order of test cases in a test suite can improve test effectiveness [1, 11-13].

Test case prioritization (TCP) involves the explicit prior planning of the execution order of test cases with the intention of increasing the effectiveness of software testing activities by improving the rate of fault detection [12, 13]. There can be many possible goals behind applying TCP, such as: to increase the rate of fault detection; to increase statement, branch or function test coverage; and/or to increase confidence in system reliability [12, 13]. To date, TCP has been primarily applied to improve regression testing efforts [11-13] of white box, code-level test cases.

Our research goal is to develop and validate a system-level test case prioritization model based upon static metrics and system failure data to improve system test effectiveness. We will build and validate the model with data from industrial products at Nortel Networks, a telecommunications company. Telecommunications systems must have high reliability so as to avoid communications failures which could potentially cause major disruptions in the daily life and workings of society. As a result, Nortel and other companies must ensure their validation and verification efforts are as effective and efficient as possible.

Nortel Networks also has a history in the development, validation, and use of predictive models, such as the one we are developing for system test prioritization. In the mid 1990's, researchers at Nortel developed the Enhanced Measurement Early Risk Assessment of Latent Defects (EMERALD) [7, 8] decision support system for assessing reliability risk with the goal of improving telecommunications software quality as perceived by the customer and end user. Trials in 1995 with a very large telecommunications system found many benefits to the use of EMERALD for guiding development activities, including testing efforts and test automation efforts were targeted more effectively [9]. Underlying EMERALD is a model based upon the static characteristics of the code. Our new model will also incorporate system failure data from system test and from the field.

2. Metrics Collection

The following candidate metrics will be included in our model building:

- Static properties of code, such as code complexity;
- Quantity of filtered (for false positives) alerts produced by one or more automated static analysis tools;

- Code churn information;
- Test coverage;
- System failure data for a component/file by prior system test efforts; and
- System failure data for a component reported by customers in the field

We will collect a significant quantity of measures for these metrics from Nortel products and examine the correlation between the metrics and the ultimate measure of the quantity of system failures that involve a particular file or component. Metrics that consistently do not demonstrate a correlation with field failures are candidates to be removed from the predictive model.

3. Model Building and Validation

We will use the metrics (as discussed in Section 2) to build a prediction model for the number of test and field failures for a module. We will examine the correlation between the actual and predicted test and filed failures. This correlation is used to quantify the sensitivity of prediction. The two correlation techniques that can be utilized are (1) the Spearman rank correlation which is a commonly-used robust rank correlation technique because it can be applied when the association between elements is non-linear; and (2) the Pearson bivariate correlation is best suited for normally distributed data and where the association between elements is linear [5]. A positive relationship between the actual and estimated values is desired in terms of the more robust Spearman rank correlation. The model will be validated if we can demonstrate a strong correlation between the prediction produced by the model and the actual results, such as within 90%-95% accuracy.

The modules will be ranked based upon the number of test and field failure predicted by the model. This ranking will then be mapped to a traceability matrix which will be used to determine the prioritization of the test cases to be run.

4. Use and Empirical Assessment at Nortel

Our empirical assessment of the efficacy of the model will involve the use of the model to prioritize the test cases of a Nortel product under development. Our primary measure for success is an increase in test effectiveness, as computed using Equation 1:

$$\text{Test effectiveness} = \frac{\text{Defects found by system test}}{\text{Defects found by system test + defects reported by customers during the first 24 months after delivery}} \quad (1)$$

Acknowledgements

This research is supported by a research grant from Nortel Networks.

References

[1] F. Basanieri, A. Betolino, and E. Marchetti, "The Cow_Suite Approach to Planning and Deriving Test Suites in UML Projects," Fifth International Conference on the Unified Modeling Language - the Language and its applications UML 2002, Dresden, Germany, September 2002, pp. 383-397.

[2] B. Beizer, *Software Testing Techniques*. London: International Thompson Computer Press, 1990.

[3] R. Craig and S. Jaskiel, *Systematic Software Testing*. Norwood, MA: Artech House Publishers, 2002.

[4] L. Crispen and T. House, *Testing Extreme Programming*. Boston, MA: Addison Wesley Pearson Education, 2003.

[5] N. E. Fenton and S. L. Pfleeger, *Software Metrics: A Rigorous and Practical Approach*: Brooks/Cole, 1998.

[6] M. Harrold, "Testing: A Roadmap," International Conference on Software Engineering, Limerick, Ireland, 2000, pp. 61-72.

[7] J. Hudepohl, S. J. Aud, T. Khoshgoftaar, E. B. Allen, and J. Mayrand, "Emerald: Software Metrics and Models on the Desktop," *IEEE Software*, vol. 13, no. 5, September 1996, pp. 56-59.

[8] J. Hudepohl, W. Jones, and B. Lague, "EMERALD: A Case Study in Enhancing Software Reliability," Eighth International Symposium on Software Reliability Engineering (Case Studies), Albuquerque, New Mexico, 1997, pp. 85-91.

[9] J. P. Hudepohl, S. J. Aud, T. M. Khoshgoftaar, E. B. Allen, and J. Mayrand, "Integrating metrics and models for software risk assessment," International Symposium on Software Reliability Engineering, White Plains, NY, 1996, pp. 93-98.

[10] IEEE, "IEEE Standard 610.12-1990, IEEE Standard Glossary of Software Engineering Terminology," 1990.

[11] J. C. Munson and S. Elbaum, "Code Churn: A Measure for Estimating the Impact of Code Change," IEEE International Conference on Software Maintenance, Bethesda, MD, 1998, pp. 24-31.

[12] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "Test Case Prioritization," *IEEE Transactions on Software Engineering*, vol. 27, no. 10, October, 2001, pp. 929-948.

[13] G. Rothermel, R. Untch, C. Chu, and M. Harrold, "Test Case Prioritization: An Empirical Study," International Conference on Software Maintenance, Oxford, UK, September 1999, pp. 179 - 188.

[14] L. Tahat, B. Vaysburg, B. Korel, and A. Bader, "Requirement-Based Automated Black-Box Test Generation," 25th Annual International Computer Software and Applications Conference, Chicago, Illinois, 2001, pp. 489-495.