

Towards a Metric Suite for Early Software Reliability Assessment

Nachiappan Nagappan, Laurie Williams, Mladen Vouk

Department of Computer Science, North Carolina State University, NC 27695

{nnagapp, lawilli3, vouk}@unity.ncsu.edu

Abstract

We are developing a suite of metrics for early assessment of software reliability and to provide feedback to the developer on the quality of their testing effort. The suite consists of easy-to-measure information collected from the source code and test programs. We are studying correlation between these metrics and the reliability of the developed software. The results of an initial case study demonstrate the feasibility of the approach. Metrics will be added to or deleted from the suite based on further validation work.

1 Introduction

An internal metric, such as the number of lines of code, is a measure derived from the product itself. An external measure is a measure of a product derived from assessment of the behavior of the system. For example, the number of defects found in test is an external measure. The ISO/IEC standard [12] states that “Internal metrics are of little value unless there is evidence that they are related to external quality.” *Our research objective is to create and validate a set of easy-to-measure internal metrics that provide feedback on the quality of a testing effort and can be used as early indication of the external measure of system reliability.* This work builds on the work reported in [15].

Structural object-oriented (OO) measurements, such as those defined in the Chidamber-Kemerer (CK) [6] and MOOD [5] OO metric suites, are being used to evaluate and predict the quality of software [11]. The CK and MOOD suites include such relatively easy-to-measure metrics as cohesion and coupling. When properly calibrated, they may be used as an early internal indicator of externally-visible product quality. The actual measurable product quality (e.g., failure rate) that derives from its behavior, usually cannot be measured until too late in the life-cycle to effect an affordable corrective action. In general, a multi-phase approach must be taken, since different metrics will be visible at different development phases [15].

For these early indicators to be meaningful, they must be related (in a statistically significant and stable way) to the field quality of the product. The validation of metric relevance requires convincing demonstration that (1) the metric measures what it purports to measure and (2) the metric is associated with an important external metric, such as field reliability, maintainability, or fault-proneness [9]. There is a growing body of empirical evidence that supports the theoretical validity of the use

of higher-order early metrics, such as OO metrics [1, 4] as predictors of field quality. However, general validity of these metrics (which, on their own, are often unrelated to the actual operational profile of the product) is still open to criticism [7]. Still, we believe that a judicious use of early metrics, in conjunction with an understanding of the software process and an early indication of the operational profile of the product, can be a powerful tool in guiding the development of good quality software.

In this paper, we report on initial results related to the development of a metric suite that would serve as early indicators of software field reliability, and that would provide feedback to the developer on the quality of the testing effort. The suite discussed here is depends on an extensive set of automated unit and acceptance test cases. Such practice is common with test-driven development approaches [3] of the Agile and Extreme Programming [2] software development methodology.

2 Metrics Suite

Our initial set of metrics called the Software Testing and Reliability Early Warning – Java (STREW-J) Version 1.1 metric suite is based on a source coverage metrics and four testing intensity ratios (Rx):

- *number of test cases/source lines of code (R1)* would indicate if there are too few test cases written to test a large body of source code;
- *number of test cases/number of requirements (R2);* would indicate if the testing was done appropriate to the requirements;
- *test lines of code/source lines of code (R3)* would indicate if the ratio R1 was inaccurate as there might have been a few test cases that might have been comprehensive;
- *number of asserts/source lines of code (R4)* would serve as a control for both R1 and R2 so that in case there are few test cases but each have a large number of successful calls to the source program then the metric suite does not penalize the developer; and on
- *code coverage (C)* which measures the ratio of the number of lines of the source code that are executed by the test code so that most of the source program is covered by testing and indicates the accuracy of R3.

The work of Vouk and Tai [15] showed that similar metrics have strong correlation with field quality of a industrial software products. Hence, while specific mix of verification and validation intensity (VVI) metrics will probably change as we further study the problem, the

concept appears to be quite a powerful early predictor of field quality.

3 Case Study

This initial study is based on 13 programs developed in a senior level software engineering course at North Carolina State University (NCSU). Measured Rx values and reliability estimates are in the Appendix. Reliability of the codes was estimated using a “no failure” model [8, 10, 13]. This model was chosen because the XP process requires that all test cases written should reflect operational usage and must pass. Hence, after they have been passed, there exists a meaningful long term failure rate denoted by Θ . N (random) tests may establish an upper confidence bound of $(1-\alpha)$ on Θ , let it be θ [10]. The relationship is given by $1 - (1-\theta)^N \leq \alpha$ [14]. So θ forms a lower confidence bound of the failure rate of our software. The result of applying an ordinary least square regression on the metrics obtained from the 13 programs and the corresponding reliability estimates yielded a statistically significant result ($F=4.325$, $p < 0.041$) on the ANOVA. The regression equation to predict the reliability was found to be:

$$Reliability = 0.669 + 1.586*R1 + 0.0513*R2 - 0.0290*R3 + 0.192*R4 - 0.0774*C$$

A correlation analysis is shown in Table 1. It indicates that there exists a statistically significant relationship between the ratios $R1$, $R2$, and the reliability estimate. There apparently exists significant relationship in the case of $R3$ and $R4$ as they depend heavily on developer characteristics. However, the relationship between the code coverage and the estimated reliability may not be statistically significant because the students were specifically assigned to create test cases with high code coverage.

Table 1: Pearson correlation and statistical significance results between the ratios and the reliability estimate

θ	R1	R2	R3	R4	Coverage
0.01	0.629, p<.021	0.992, p<.000	0.302, p<.316	0.469, p<.106	0.118, p<.702
0.05	0.703, p<.007	0.814, p<.001	0.393, p<.184	0.541, p<.056	0.039, p<.900
0.10	0.616, p<.025	0.584, p<.036	0.414, p<.159	0.502, p<.081	-0.071, p<.817

4 Conclusions and future work

This work in progress reports on development of a set of early software field quality indication metrics. The results of an initial case study that assessed the relationship between the metrics and the reliability estimates demonstrate the feasibility of the approach and the candidate metrics. However, validation of the approach will require data collection and field reliability analysis in a multitude of environments. We plan an extensive validation of our metric suite in a variety of

different industrial and academic environments. We will conduct another academic experiment in the Fall 2003 semester and will collect data to analyze the reliability growth based on the failures encountered during testing. We will also assess “true” field reliability of the programs against a uniform set of test cases. Through this work, we will refine the metric suite by adding and deleting metrics based on the results of these case studies.

5 Acknowledgements

This work was supported in part by an IBM-Eclipse Innovation Award, IBM SUR Grant, NC State Center for Advanced Computation and Communication, and NSF award #9901004.

References

- [1] V. Basili, L. Briand, and W. Melo, "A Validation of Object Oriented Design Metrics as Quality Indicators," *IEEE Transactions on Software Engineering*, vol. 22, pp. 751-761, 1996.
- [2] K. Beck, *Extreme Programming Explained: Embrace Change*. Reading, Massachusetts: Addison-Wesley, 2000.
- [3] K. Beck, *Test Driven Development -- by Example*. Boston: Addison Wesley, 2003.
- [4] L. Briand, K. El Emam, and S. Morasca, "Theoretical and Empirical Validation of Software Metrics," 1995.
- [5] F. Brito e Abreu, "The MOOD Metrics Set," presented at ECOOP '95 Workshop on Metrics, 1995.
- [6] S. R. Chidamber and C. F. Kemerer, "A Metrics Suite for Object Oriented Design," *IEEE Transactions on Software Engineering*, vol. 20, 1994.
- [7] N. I. Churcher and M. J. Shepperd, "Comments on 'A Metrics Suite for Object-Oriented Design'," *IEEE Transactions on Software Engineering*, vol. 21, pp. 263-5, 1995.
- [8] W. Ehrenberger, "Statistical Testing of Real-Time Software," in *Verification and Validation of Real-Time Software*, W. Quirk, J., Ed. New York: Springer-Verlag, 1985.
- [9] K. El Emam, "A Methodology for Validating Software Product Metrics," National Research Council of Canada, Ottawa, Ontario, Canada NCR/ERC-1076, June 2000.
- [10] D. Hamlet, Voas J., "Faults on Its Sleeve: Amplifying Software Reliability Testing," presented at International Symposium on Software Testing and Analysis, Cambridge, MA, 1993.
- [11] R. Harrison, S. J. Counsell, and R. V. Nithi, "An Evaluation of the MOOD Set of Object-Oriented Software Metrics," *IEEE Transactions on Software Engineering*, vol. 24, pp. 491-496, June 1998.
- [12] ISO/IEC, "DIS 14598-1 Information Technology - Software Product Evaluation," 1996.
- [13] K. W. Miller, Morell, L.J., Noonan, R.E., Park, S.K., Nicol, D.M., Murrill, B.W., Voas, J.M., "Estimating the Probability of Failure When Testing Reveals no Failures," *IEEE Transactions on Software Engineering*, 1992.
- [14] R. Thayer, Lipow, M., Nelson, E., *Software Reliability*. Amsterdam: North-Holland, 1978.
- [15] M. A. Vouk, Tai, K.C., "Multi-Phase Coverage- and Risk-Based Software Reliability Modeling," presented at CASCON '93, 1993.