Advanced and Applied Convergence Letters AACL 04

Advanced and Applied Convergence

1st International Joint Conference, IJCC 2015
Ho Chi Minh City, Vietnam, February 2015
Revised Selected Papers

IIBC
IPACT

Table of Contents

A Study of Software Defect Rate Estimation Technique in Complete Repeat Testing Environments

Young B. Park*, Mahmoud Tarokh**, and R. Young Chul Kim***

*Dankook University, Chonan, Korea
e-mail : ybpark@dankook.ac.kr
**San Diego State University, San Diego, USA
***Hongik University, Sejong Korea
e-mail : bob@selab.hongik.ac.kr

Abstract

Software testing is performed to find software defect, but it is always not enough tests are made to test all the cases. Re-testing such as regression test, cannot grantee software quality. It only provides confidence on the changed code. In this paper a defect estimation technique is proposed. Using this technique, number of re-testing can be calculated to achieve target defect rate. This technique can be applied to the automated testing via software testing tool, it will shows when the test stops.

Keywords: *Complete Repeat Testing, Defect Rate Estimation, Regression Test.*

1. Introduction

Testing is a complex and cost taking activity which requires time and resources[2]. Re-testing-all is a basic re-testing mechanism. Even though re-testing-all mechanism is applied, there is no way to guarantee the defect rate of the software. Re-testing subject has been studied in the field of repeat inspection. In 1980's, Raouf, Jain, and Sathe were the first one to develop a model for determining the optimal number of repeat inspections[3]. Raz and Thomas proposed the inspection line with a series of multiple inspections[4]. Drury, Karwan, and Vanderwarker examined different ways of combining the results of two different sequential inspections using dynamic programming[1]. Tang provided a rule for determining the optimal sequence of inspection[5]. Repeat inspection and re-testing has a same back ground; by the mean of repeating tests, the quality of the target can be improved. With proposed framework the quality of target can be estimated.

2. Problem Modeling

In the software testing, test is performed when the iteration or project stage is finished. Once the code is tested, all the testing results are reported to the developer team for rework. After the developer team performs debugging work, re-test the software until it meets the target quality. This process can be modeled as follow (Figure 1);

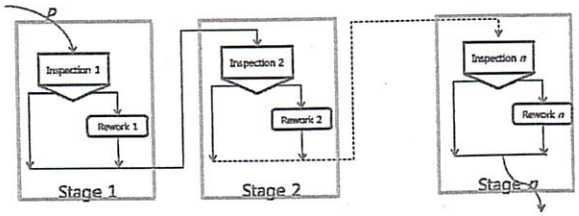

Figure 1. Complete Repeat Testing

J.J. Kang et al. (Eds.): IJCC 2015, AACL 04, pp. 34~36, 2015

where P is an initial probability of the software being defective. Each stage has two components, one is inspection and the other one is rework. Let's $P_i$is a given probability of being defective when the entering i th stage. And let P_i' is a residual probability of being defective after i th rework. Figure 2 shows each stage;

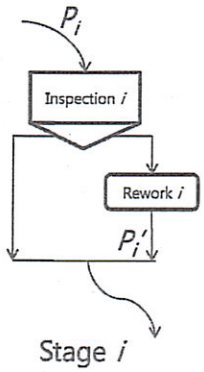

Figure 2. i'th Single Stage

In this paper, it is assumed that there are n testing stages for n inspector. Let's say there are n Inspectors$\langle I_1, I_2, \cdots, I_n \rangle$.

3. Experimental Analysis

There are two kinds of human errors (decision mistakes) in an inspection; the one is false decision of being defective which is not defective (Type I error) and the other one is false decision of being non-defective which is defective (Type II error). Falsely reject means the former (Type I error) and falsely accept means the latter (Type II). Let's say Pri is a probability of falsely reject for inspector I_i and Pai is a probability of falsely accept for inspector I_i. In software testing, since the software testing is performed by automatic testing tools, Pri are relatively small (it is most unlikely making false decision). But since it is impossible to cover all kinds of testing, Pai can be relatively large. Usually Pai is associated with testing coverage. In this paper, we assumed Pri = ξ (where ξ is small constant). Table 1 shows the probabilities of being "non-defective" and "defective" within "Accept" and "Reject"

Table 1. Probability Table

	Accept	Reject
Non-defect	$(1 - P_i)(1 - Pr_i)$	$(1 - P_i)\ Pr_i$
Defect	$P_i \cdot Pa_i$	$P_i \cdot (1 - Pa_i)$

Now, consider out-quality after the inspection. The probability of being defective after inspection is as follow;

$$P_o = \frac{P_i \times Pa_i}{(1 - P_i)(1 - Pr_i) + P_i \cdot Pa_i}$$

After rework or debugging, it contains defection in the software. The probability of rework is same as probability of the reject;

$$Reject = (1 - P_i) \cdot Pr_i + P_i \cdot (1 - Pa_i)$$

Since P_i' is a residual probability of being defective, the probability of being defective can be achieved by multiplying P_i' with Reject.

$$P_o' = P_i' \cdot (P_i \cdot (1 - Pa_i - Pr_i) + Pr_i)$$

Since the result of i th stage is the input of i+1th stage, after rework, the probability of begin defective can be archived as follow;

$$P_{i+1} = P_i \cdot Pa_i + P_i' \cdot (P_i \cdot (1 - Pa_i - Pr_i) + Pr_i)$$

Final quality of software can be calculated as fallow;

step 1 set $P_i = P$, and $i = 1$

step 2 calculate
$P_o = P_i \cdot Pa_i + P_i' \cdot (P_i \cdot (1 - Pa_i - \xi) + \xi)$

step 3 set $P_i = P_o$, and $i = i + 1$

step 4 repeat step 2 and step 3 until $i = n$

step 5 submit P_i as a result

4. Conclusion

As software is getting complex, it is very difficult to perform enough test. As a result automated testing tools are introduced in the software development process. Since an automated testing tool can test automatically, repeat testing is much convenient than ever. But testing coverage is another matter, automated testing tool cannot design testing model, as a result, after testing it is not easy to tell how much defects are left in the software. In this paper a defect estimation model is proposed and demonstrated its usage. Along with automated software testing tool, this technique can provide visible defect rate estimation.

5. Acknowledgement

This research was supported by Next-Generation Information Computing Development Program through the National Research Foundation of Korea (NRF) funded by the Ministry of Science, ICT & Future Planning (No. 2012M3C4A7033348) and the ICT R&D program of MSIP/IITP. [10044457, Development of Autonomous Intelligent Collaboration Framework for Knowledge Bases and Smart Devices]

6. References

[1] Drury, C. G., Karwan, M. H. and Vanderwarker, D. R., "The two-inspector problem", IIE Transactions, 18/2, 174-181, 1986.

[2] H. Do, S. Mirarab, L. Tahvildari, G. Rothermel, "The effects of time constraints on test case prioritization: A series of controlled experiments," Journal of IEEE Transactions on Software Engineering, vol.36, no.5, pp.593-617, Sep/Oct. 2010.

[3] Raouf, A., Jain, J.K., and Sathe, P.T., "A cost minimization model for multicharacteristic component inspection", IIE Transactions, 15, pp. 187–194, 1983.

[4] Raz, T. and Thomas, M.U., "Method for Sequencing Inspection Activities Subject to Errors", IIE Transactions, 15(1), pp. 12-18, 1983.

[5] Tang, K., "Economic Design of Product Specifications for a Complete Inspection Plan", International Journal of Production Research, 26(2), pp. 203-217, 1987.