Do the software reliability prediction models serve their intended purpose?

Yuan Wei, Medtronic, Inc. *Email: yuanweiumd@gmail.com*
Jin Qin, University of Science and Technology of China, Email: qjin.elgoog@gmail.com

Issues on software reliability prediction models

The reliability of software is increasingly emphasized as more and more functions, which were formerly provided by hardware, are highly dependent on software. Many techniques or methodologies have been published to prevent fault occurrences, to detect the faults and remove them more effectively and efficiently, to tolerate the faults by providing redundant service, and to estimate or predict the fault occurrence via software reliability modeling. Those techniques, more or less, have been tried in different industries. No matter what techniques for fault detection, removal, and tolerance are applied, key questions are often asked about the readiness of release from quality or reliability standpoint. Is the software bug free or failure rate is acceptable? Most of time, pass of all test cases (or some threshold is met) is used as one key criteria as well as other factors like budget, market strategy, etc. The quantitative reliability question, the probability of failure-free execution for a specified time in a specified environment without any failures, have not played a critical role in decision making. Researchers have proposed multiple models [1] ~ [5]. Those models use testing data on software to estimate how reliable the software is now and predict the reliability in the future. Even the models show promising results with their own experimental data, mixing results come up with broader applications [6] [7]. That makes the development organization question if the prediction models serve their intended purpose.

Key reasons for un-fit models

There are many reasons which could make the results from prediction models far away from the reality. The assumptions used by the models are not always true in reality [8]. Based on authors' experience, the top contributors include:

- ◊ Operational profile
- ◊ Testing strategy
- ◊ Resource allocation

Operational profile

The operational profile is a quantitative characterization of software usage by customers. It is very difficult, costly and even impossible to obtain the real field operational profile, especially for new software products without any historical information. There is always a gap between the testing profile used by testing engineers and the real operational profile. Also, the operational profile could help better select test case while it does not contribute to the prediction accuracy significantly.

Testing strategy

The failure rate in software has strong relationship with the complexity of the software. The higher the complexity, the larger the failure possibility. It is reasonable to expect that more bugs will be found on the software components / features with high complexity. A jump in the failure rate trend sometimes represents high complexity modules are being tested. Different companies or organizations, or different projects in the same organization have different testing strategies.

Some teams prefer to design and test the software components with high complexity in the early phase so there will be enough time to fix the bugs . Others want to focus on those modules in the middle phase when more knowledge on the high complex components is obtained to avoid introducing more bugs and design better test cases.

Resource allocation

The testing resource is not allocated evenly during the whole testing phase. Less resource may be assigned in the early phase and more resource may be added in later phases. Hence, it is reasonable to detect more failures with more test efforts. Also, during the development phase, resource may be reallocated for companies' other considerations like priority, schedule, market release strategy, and etc. The resource allocation directly impacts the curve of the failure detection.

Solution?

There are no clear solutions available by date. Academic and industry have their own concerns or challenges on finding a widely applicable prediction model. Academic does not have sufficient funding to validate the models in various applications while industry does not think it particularly worthwhile due to no high degree of confidence on the prediction results. Positive feedbacks from the industry applications are also seen for improving prediction results of software reliability growth models. To reach that goal, an organization first should have a mature system and software development processes. That will ensure consistent development results from various projects. Second, normalize the failure data by testing efforts. Third, apply the models after system integration complete. The last, validate your models with several release before adopting it for decision making.

Reference:

[1] Moranda, P.B., "An Error detection model for application during software development," *IEEE Trans. Reliability*, Vol. R-28(5), 1979

[2]] Cai, K.Y., "On estimating the number of defects remaining in software," *Journal of Systems and Software*, Vol, 40(1), 1998

[3] Musa, J.D. And K. Okumoto, "Application of basic and logarithmic Poisson execution model in software reliability measure," in *The Challenge of Advanced Computing Technology ot System Design Method*, NATO Advanced Study Institute, 1985

[4] Ohba, M. and S. Yamada, "S-Shaped software reliability growth models," in *Proc. 4th International Conference on Reliability and Maintainability*, Perros Guirec, France, 1984

[5] Pham, H. And X. Zhang, "An NHPP software reliability model and its comparison," *Int. J. Reliability, Quality and Safety Engineering*, Vol. 4(3), 1997, 269-282.

[6] Wei, Y., "Improve the Prediction Accuracy of Software Reliability Growth Model," *21st IEEE International Symposium on Software Reliability Engineering (ISSRE),* San Jose, CA, Nov. 1st ~ 4th, 2010

[7] Chakrabarti, S. and Kuma, P., "Software Reliability Prediction in Philips Healthcare – An Experience Report", *20th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, Mysuru, India, Nov. 16th ~ 19th.

[8] Wood, A., "Software reliability growth models: assumptions vs. reality,"", *8th IEEE International Symposium on Software Reliability Engineering (ISSRE)*, pp 136-141, Albuquerque, NM , USA, Nov. 2nd ~ 5th, 1997

[9] Ji, Y., Mookerjee, V., and Sethi, S., "Optimal Software Development: A Control theoretic Approach," *Information Systems Research*, pp. 292-306, Vol. 16. No. 3 Sept. 2005