A New Approach for Software Testability Analysis *

[Extended Abstract]

Liang Zhao
Department of Computer Science & Technology, TsingHua University
Beijing, China, 100084
zhaoiang02@mails.tsinghua.edu.cn

ABSTRACT
Software testability analysis has been an important research direction since 1990s and becomes more pervasive when entering 21st century. In this paper, we summarize problems in existing research work. We propose to use beta distribution to indicate software testability. When incorporating testing effectiveness information, we theoretically prove that the distribution can express testing effort and test value at the same time. We conduct experiment and validate our results on Siemens programs. Future work concentrate on deducing a prior estimation of the distribution for given software and testing criterion pair from program slicing and semantic analysis.

Categories and Subject Descriptors
D.2.5 [Software Engineering]: Testing and Debugging—Testing Tools; D.2.8 [Software Engineering]: Metrics—Process metrics, Product metrics

General Terms
Measurement, Design, Reliability

Keywords
Software testability; testing effectiveness; measure; beta distribution

1. RESEARCH PROBLEM
Testability is a distinct software quality characteristics. It expresses the affect of software structural and semantic on the effectiveness of testing following certain criterion. The effectiveness of testing decides the quality of released software. Testability is an elusive concept. It is difficult to get a clear view on all the potential factors that can affect testability and the dominant degree of these factors under different testing context.

The research on software testability first appeared in 1975[1]. It is adopted in McCall [2]and Boehm [3] software quality model, which build the foundation of ISO 9126 quality model [4]. Since 1990s, software engineering community began to launch quantitative research on testability. Although some achievements have been made on testability measure [5, 6, 7, 8, 9, 10, 11, 12, 13], design for testability [14, 15] and applying testability to reliability estimation [16, 17], these progresses have not been widely adopted in software practice. What makes this seemed important issues out of the usability of software industry? What are the limitations of existing research? These can be manifested in following 4 aspects:

1. The confusion of testability definitions. There are many definitions of software testability. IEEE [18] defines it as the degree to which a system or component facilitates the establishment of test criteria and performance of tests to determine whether those criteria have been met. ISO [4] defines it as attributes of software that bear on the effort needed to validate the software product. These two standard definitions aim to different targets and both are qualitative without any operational guidelines. Many researchers proposed their own definitions to proceed analysis, such as, observability and controllability [6], “prediction of the tendency for failures to be observed during random black-box testing when faults are presented” [8], “potential conflict that may occur during test” [9], “the effort to test the program according to certain criterion” [13], and so on. These definitions reflect the nature of testability from different points of view, but also introduce chaos of understanding on testability.

2. Internal attributes or external attributes? Testability is explicitly defined as external attributes in [19], which correlates with certain testing criterion and testing context. Most research take this view, such as PIE[10], fault tree analysis[13], control-flow based analysis [5] etc.. Some still implicitly analyze it as an internal attributes, such as DRR[8], domain testability[6], class interactions [9], and other data flow based analysis. Fenton et al. points out that "Testability is arguably as external attributes"[19]. Testability is by definition linked with testing [20]. Testing effectiveness is not only affected by the features of particular software,
but also by testers, test tools and test context. Although internal attributes are the decisive factors that lead to the variation of external behavior, the main testability research issue should lay more emphasis on the interaction between software and testing criterion.

3. Separation of concerns. Separation of concerns is a best practice for software design and implementation, but not for software analysis, which needs a holistic view of software. Testing can achieve two main goals, reliability improvement and reliability assessment. The key feature of testing is its ability to reveal faults. Testing is of less value if it fails to detect existing problems no matter how economical it is. It is intuitively true that more testable software can achieve particular reliability goal with less test effort. Testability analysis is an essential part of in test effectiveness assessment. There are two focuses of testability analysis, viz. ease of performing testing and value of testing [14]. The former reflects the effort to performing testing according to certain criterion. The latter concerns the effectiveness of testing. Ideal testability measure should indicate both these properties while existing research separately analysis these two focuses and can’t give a comprehensive view.

4. Problem of compositionality. Compositionality is one of the research challenge faced by software engineering [21]. It is also the same for testability analysis. There are no systemic or formal way to derive system’s testability metric from its component’s. According to some existing testability analysis model, e.g. DRR metric, the system may be more testable than its components, which means we may combine two untestable subprograms into a testable one[22]. This violates the measure principle [19] and doesn’t conform to the reality.

2. RESEARCH HYPOTHESIS

Being an external attributes, the basic testability research subject should be software and test criterion pair. It tries to use software information and test criterion’s requirement to predict the achieved goal. The traditional single value measure method has shown more limitations. Researcher begin to introduce more powerfully interpretative method to analyze software testability [23, 24]. Following IEEE definition, we propose to use distribution to indicate software testability.

Assume test failures are statistically independent bernoulli trials. Let $\theta$ be the probability of failure. We can represent the prior distribution of $\theta$ as $\beta(a, b)$. Within Bayesian framework, when the prior and posterior distribution belong to the same parametric family of distributions, it can represent a kind of homogeneity in the way in which man’s beliefs change as we receive extra information [25, 26]. In this sense, beta distribution is a good candidate conjugate distribution family.

Given certain software reliability goal, expressed as failure rate ($\theta_0$) and corresponding confidence level ($C$), according to the Bayesian statistical method, the least number of test that needed to assure the announced reliability goal should satisfy:

$$\int_0^{\theta_0} \frac{\theta^{a-1}(1-\theta)^{b+N-1}}{B(a,b+N)} \, d\theta > C. \quad (1)$$

Above formula can indicate the quality of testing ($\theta_0, C$) and the quantity of testing ($N$) simultaneously, which are decided by the parameters $a$ and $b$. [27] evaluated the achieved reliability when testing reveals zero fault. [26] studied the necessary test effort for certain reliability goals when testing reveals different number of faults. Their study mainly considered classical statistical testing and did not study the effect of testing when following different test criterion.

Both [27] and [26] admitted that when incorporating dynamic test and static software information into the model, it can produce more precise result. Our research objective is try to introduce testing effectiveness information into the model and to get prior distribution estimation for each software and test criterion pair.

3. PROPOSED SOLUTION

The work can be divided into two parts to achieve the research objective. The first part of work is to validate the reasonableness of beta distribution as testability indication. Then we can try to find ways to deduce the distribution for each software and test criterion pair.

To validate appropriate beta distribution for specific program and testing criterion pair we follow three main steps.

The first step is to define appropriate testing criterion’s effectiveness measure. The stable and objective of the measure is the basis for other work. The second step is to introduce criterion’s effectiveness information into the model. Although parameter estimation is a classical mathematical problem, it must face the unique situation in software engineering, where more subjective factors and randomness are included in the sampling process.

The third step is to prove that when incorporating effectiveness information, the distribution can provide reasonable estimation on the quantity and quality of testing.

4. CURRENT PROGRESS

4.1 Testing Criterion’s Effectiveness Measure

Usually P-measure (probability of detecting at least one fault) and E-Measure (the expected number of faults to be detected) are used as criterion’s effectiveness indication [28, 29]. These two measures give no help for software reliability estimation. To introduce testing effectiveness information in software reliability assessment and reduce the randomness and subjectiveness in effectiveness measure, we propose $fdp$ as a new effectiveness measure.

Fault Detection Probability: Fault Detection Probability ($fdp$) is the ratio of suites satisfying specific test criterion that can detect particular fault.

$$fdp = \frac{\text{Num. of suites that can detect Fault } F}{\text{Total Num. of Test Suites Satisfying Criterion}}$$

$fdp$ is a function of software, fault and test criterion. It can give a more stable and comprehensive view of testing. In theory, there are infinite number of suites satisfying certain testing criterion. In practice, when the suite number is big enough, we can get a good $fdp$ approximation for different faults included in the program. Using them as sample data, we can get the parameter estimation according to classical statistical estimation method, such as Maximum Likelihood Estimation (MLE), least squares error estimation...
than 1. These inconsistency originates from the ability of result are bigger than 1, while average testing criterion pair. The results suggest that:

\[ \text{MLE} \]

\[ \text{branch estimation and inequa-} \]


4.2 The Express ability of the Beta Distribution

Inequation 1 defines N as the implicit function of a and b. In preliminary study we have proven that N is increasing in a with the slope 

\[ \frac{\ln a - \ln b}{\ln (1 - \xi) - \ln (1 - \eta)} \]

\[ \xi \in (0, \theta_0), \eta \in (0,1) \text{ and } \xi \neq \eta \] and is decreasing in b with the slope -1. So parameter a of the distribution can express the effect of all factors that have negative impact on testability and parameter b can express the effect of all factors that have positive impact on testing.

4.3 The Experiment

To validate the usability of the model, we conduct experiments on well known Siemens programs. For each faulty version, we test it with two different testing criteria, branch coverage testing and corresponding random testing. Using achieved fdp data, we get the distribution for each program and criterion pair. The result is presented in Table 1.

The result shows that the difference originated from three estimation method is small. According to the mathematical property of beta distribution, '1' is the threshold [25]. Any parameters vary from less than to bigger than '1' will change the shape of distribution. For random testing of program printtoken1 and branch coverage testing of schedule2, \( \sigma^2 \) based estimation and \( s^2 \) based estimation are less than 1, while MLE result is bigger than 1. For branch coverage testing of program schedule1, \( \sigma^2 \) based estimation and MLE result are bigger than 1, while \( s^2 \) based estimation is less than 1. These inconsistency originates from the ability of different statistical methods [25].

According to \( \sigma^2 \) based parameter estimation and inequation 1, we get the least value of N to assure given reliability goal (0.001,0.99). The result is presented in Table 2. The row \( a=1 \ b=1 \) is the estimated test number ignoring the program and test information. The random testing is the result from the estimated distribution for program and random testing pair, and the row branch coverage is the result from the estimated distribution for program and branch coverage testing criterion pair. The results suggest that:

- Inequation 1 produces the same result with the classical statistical method when the parameter \( a=1 \) and \( b=1 \) [30]. When incorporating testing effectiveness information, the results vary for different program and test criterion pair.
- Branch coverage testing and random testing play different effectiveness on different program.
- Generally branch coverage testing is more effective than random testing but not always so. There should be a fitness function between subject software and certain testing criterion.

So when (program, criterion) is more testable, it needs less test effort to achieve reliability goal. When (program, criterion) is less testable, it asks for more test effort. Classical statistical software reliability assessment method is a ‘blind’ method to a certain degree. Sometimes it overestimate the test effect. Potential risks may stem from the optimistic inflation. Some times it underestimate the test effect and produce more conservative result, which will waste limited test resource.

Our preliminary conclusion is that the estimated distribution does reflect the effectiveness of different testing criterion on different program more precisely than prediction ignoring test effectiveness information. The distribution can give a provable improvements of one testing criterion over another and lays the foundation for future work.

5. FUTURE WORK AND CONCLUDING REMARKS

Our work focus on the interaction between software and test criterion. From theoretically analysis and experiment validation, we have get a hinder testability indication for software and criterion pair. From another point of view, software failure resulted from different fault included in the software. Some researchers try to estimate the fault prone-ness [31] and fault number from the software systems[32]. These research have achieved some preliminary results. Dif-ferent faults have different test difficulty [33]. If we can predict the failure rate caused by different fault under different testing criterion, we can get the prior estimation of the model’s parameters, i.e. a prior indication of software testability, and predict necessary testing effort for specified software (test) quality.

6. REFERENCES

Table 2: The Necessary Test Number for Given Reliability Goal (0.001, 0.99)

<table>
<thead>
<tr>
<th>criterion</th>
<th>schedule1</th>
<th>schedule2</th>
<th>printtoken1</th>
<th>printtoken2</th>
<th>replace</th>
<th>totinfo</th>
<th>teas</th>
</tr>
</thead>
<tbody>
<tr>
<td>a=1 b=1 branch testing</td>
<td>4602</td>
<td>4602</td>
<td>4602</td>
<td>4602</td>
<td>4602</td>
<td>4602</td>
<td>4602</td>
</tr>
<tr>
<td>random testing</td>
<td>3463</td>
<td>7005</td>
<td>4450</td>
<td>7685</td>
<td>4771</td>
<td>4465</td>
<td>4449</td>
</tr>
<tr>
<td></td>
<td>4613</td>
<td>4461</td>
<td>3037</td>
<td>4298</td>
<td>4305</td>
<td>3658</td>
<td>3479</td>
</tr>
</tbody>
</table>


988