Test Coverage Dependent Software Reliability Estimation by the HGD Model

Raymond Jacoby * and Kaori Masuzawa

TOSHIBA Corporation,
Systems & Software Engineering Laboratory
Yanagi-cho, Saiwai-ku, Kawasaki-shi, 223 Japan
E-mail: { jacey | kaori } @ssel.toshiba.co.jp

Abstract

The Hyper-Geometric Distribution Model (HGDM) has been presented as a software reliability growth model with the capability to make estimations for various kinds of real observed test-and-debug data. With the HGDM, the discovery and rediscovery of faults during test-and-debugging has been discussed. One of the parameters, the "ease of test" function w(i) represents the total number of faults discovered and rediscovered at a test instance.

In this paper, we firstly show that the "ease of test" function w(i) can be expressed as a function of the test coverage for software under test. Test coverage represents a measure of how much of the software has been tested during a test run. Furthermore, the ease of test function w(i) can integrate a user experience-based parameter c that represents the strength of test cases to discover faults. This parameter c allows the integration of information on the goodness of test cases into the estimation process.

The application of the HGDM to a set of real observed data clearly shows that test coverage measure can be integrated directly into the "ease of test" function w(i) of the model.

Index Terms— hyper-geometric distribution model, HGDM, test coverage, initial number of software faults, test-and-debugging, user experience-based reliability estimation.

1 Introduction

In our past publications [8, 9, 2, 4], we have presented HGDM, the Hyper-Geometric Distribution Model, as a software reliability estimation model. The basic background theory has been given in [8], the basic mean value function has been established precisely in [4]. We also showed the applicability of the HGDM to various kinds of real observed test-and-debug data of software. It was shown that the HGDM has the capability to make estimations for both, exponentially growing real observed cumulative bugs curves, as well as for those showing as S-shaped behavior. Therefore, the relationship of the HGDM to other software reliability models, especially NHPP models has been made clear in [9] and [2]. With the application of one single model to all kind of real observed test-and-debug data, we do need to worry about which model is more appropriate when considering the goodness-of-fit. In [4] and [10], methods for calculating the parameter values of the model have been presented.

Also, considering the difference between failures and faults, a specific HGDM Testing Environment, which reflects the basic theory of the hyper-geometric distribution, has been set up for the HGDM.

In this paper, the main emphasis is put on the ease of test function w(i) of the model which integrates a test coverage measure. Generally, coverage is defined and measured with respect to a specific criterion, such as branch coverage, path coverage, statement coverage, etc. At the beginning stages of test, since many faults remain in the software, test coverage might be low. With the removal of these discovered faults, input test data sets might increasingly cover larger parts of the software under test. If at a final test, the whole software would be test-covered, then a large amount of the total number of initial faults are rediscovered. The total amount of faults might not be rediscovered. A faulty software path might be activated by some input data, but due to the nature of the selected input data, the corresponding fault is not discovered.

With the application of HGDM, we can estimate not only the total number of initial software faults by taking into account the software coverage. But also, considering the project-dependent, experience-based goodness (strength, ability) of test instances, the test coverage can be estimated. This "experience-based goodness" for test instances is a parameter that indicates how well the test instances perform to discover faults.

The advantages of the application of HGDM are: (1) its applicability to various real observed test-and-debug data, (2) the possibility to integrate test-progress dependent information, and (3) its consideration of test coverage knowledge at test instances.

In this paper, first the basic theory of the HGDM as a software reliability estimation model will be stated. Next, we discuss test coverage measures and the integration of such test coverage information into the HGDM Model. In the fourth section, a set of real ob-
served test-and-debug data will be explained. This
data set includes various information about the test-
and-debug process, specifically the amount of software
that has been covered at each test instance. Fifth, we
show the applicability of the HGDM to this data, espe-
cially including the information about the test cover-
age into the model. Furthermore, estimates of HGDM
will be compared to those obtained by other software
reliability estimation models. A last section concludes
our results obtained and explains about future research
activities.

2 HGDM - SRE Model

In this section, the basics of the Hyper-Geometric Dis-
tribution Model (HGDM) as a software reliability es-
timation model are briefly stated. The model's mean
value equations are explained, and the ease of test func-
tion w(i) discussed precisely.

2.1 Basic Theory of the HGDM

During the design and implementation phases, pro-
grammers have developed and debugged their software
product. With the confidence that no many faults are
left in it, the system is then forwarded to the testing-
team that performs rigorous test-and-debugging. Usu-
ally, more faults are discovered by these test-workers.

The HGDM was applied to argue the discovery and re-
move (fixing) process of faults at the test-and-debug
stage [8, 9, 2].

At the beginning of the test-and-debug stage, m ini-
tial faults (bugs) are resident in the software system.

With the application of test instances t(i), (i.e. a day
of test, a week of test, ...), errors, caused by faults that
are in the program, manifest themselves as failures that
are observed by the test-workers. A test instance t(i)
consists of test cases (i.e. various test runs, ...), that
represent the basic fault discovery mechanism. At the
end of a test instance t(i), each fault will be classified
into one of the two following categories of newly dis-
covered or previously discovered faults. In this
way, a "Total Resume Bug List (i)" contains the num-
ber of faults that are discovered and rediscovered upon
the application of a test instance t(i). This number of
faults is denoted as w(i). In [4], we have introduced a
specific HGDM testing environment for this capture-
recapture process.

w(i) is called the ease of test function and is a
key measure in the basic theory for the HGDM. For the
first test instance t(1), the number of discovered faults
is, of course, w(1). On the application of t(2), the
number of newly discovered faults is not necessarily
w(2), because some of these w(2) faults may have been
discovered by t(1). Such values w(i) are the input data
for HGDM.

Considering the application of a test instance t(i),
let C(i - 1) be the cumulative number of faults newly
discovered by t(1), t(2), ..., t(i - 1) and x be the num-
ber of faults newly discovered by t(i). C(0) is defined
to be 0. Then, some of w(i) faults discovered by t(i)
may be those that are already counted in C(i - 1), (y
rediscovered faults ), and the remaining of w(i) faults
accounts for the newly discovered faults, (z). This ba-
sic idea is depicted in Fig. 1.

\[
\begin{align*}
\text{x} &= \text{number of newly discovered faults} \\
\text{y} &= \text{faults that have been discovered in previous test instances t(j), } j = 1 \text{ to } i - 1. \\
m - C(i - 1) &= \text{not discovered faults} \\
C(i - 1) &= \text{faults discovered and rediscovered by t(i)} \\
w(i) &= x + y
\end{align*}
\]

Figure 1: Basic Idea for the HGD Model

With the assumption that the software environment
does not vary during the testing and debugging phase,
and also that the fixing process does not introduce ad-
ditional faults in the program, software reliability is
growing with the discovery and fixing of faults as test-
and-debugging proceeds.

2.2 Assumptions for the HGDM

Considering the distinction between the discovery of
faults, the detection of errors, and the observation
of failures at the application of test instances t(i), the
Hyper-Geometric Distribution Model for the estima-
tion of the number of initial faults has been proposed in
[8]. The following are the assumptions for the Hyper-
Geometric Distribution Model.

1. Faults which have been discovered upon the appli-
cation of a test instance t(i) don't have to be re-
moved (fixed), before the following test instance
t(i + 1) is applied. These faults, Not yet fixed in
t(i), might be rediscovered later.

This assumption lessens the rigorously strict as-
sumption of some other models, [6], that discov-
ered faults must be removed prior to next test oc-
casions. For difficult bugs, this removal might be
a hard task.

2. During the removal of discovered faults, no faults
will be inserted newly. The reliability of the pro-
gram is growing along with the progress in test-
and-debugging.\(^1\)

3. w(i) faults discovered by a test instance t(i) are
those taken randomly out of m initial faults. Those
w(i) may or may not have been discovered in
previous test instances.

4. w(i) discovers some of initial m. Therefore,
w(i) is represented as a function of m and the
progress in test-and-debugging p(i), (i.e. p(i) =
\text{skill of tester}, [9], [2]). With w(i) \leq m,

\[
w(i) = m \ast p(i).
\]

\(^1\)This assumption is subject to criticism. The modification of
this assumption to match a more realistic test-and-debug envi-
noment is subject to next research activities.
2.3 Mean Value Function of the HGDM

In [2], we have shown the exact development of the model in terms of mathematical equations. The mean value function of the HGDM is given by

\[
E[C(i)] = E[m] \cdot \left[ 1 - \sum_{j=1}^{i} \left( 1 - \frac{w(j)}{E[m]} \right) \right]
\]

or (2)

\[
E[C(i)] = E[m] \cdot \left[ 1 - \sum_{j=1}^{i} (1 - p(j)) \right]
\]

\(\forall i = 1 \ldots n\) test instances, with \(E[C(0)] = 0\). \(E[C(i)]\) denotes the estimated cumulative number of newly discovered faults at a test instance \(t(i)\), and \(E[m]\) the estimated total number of initial faults.

The curve for the number of newly discovered faults can be formulated as [4]

\[
fn(i) = \left[ \prod_{j=1}^{i-1} (1 - p(j)) \right] \cdot w(i) \tag{3}
\]

\(\forall i = 2 \ldots n\), with \(fn(0) = 0\) and \(fn(1) = w(1)\).

Furthermore, the curve for the number of faults rediscovered can be given by

\[
fr(i) = w(i) \cdot \left( 1 - \prod_{j=1}^{i-1} (1 - p(j)) \right) \tag{4}
\]

\(\forall i = 2 \ldots n\), with \(fr(0) = 0\) and \(fr(1) = 0\).

2.4 \(w(i)\) - Ease of Test Function

\(w(i)\) is a measure that represents how many faults have been discovered and rediscovered at a test instance \(t(i)\). This number of faults is unknown and therefore needs to be estimated.

<table>
<thead>
<tr>
<th>(w(i))</th>
<th>(w(i) = E[m] \cdot p(i))</th>
<th>(i = ) test instance (t(i))</th>
<th>(E[m])</th>
</tr>
</thead>
<tbody>
<tr>
<td>(w(i) = E[m] \cdot p(i))</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
<tr>
<td>(w(i) = E[m] \cdot )</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
<tr>
<td>(w(i) = E[m] \cdot )</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
<tr>
<td>(w(i) = E[m] \cdot )</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
<tr>
<td>(w(i) = E[m] \cdot )</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
<tr>
<td>(w(i) = E[m] \cdot )</td>
<td>(i = ) test instance (t(i))</td>
<td>(E[m])</td>
<td></td>
</tr>
</tbody>
</table>

Table 1: \(w(i)\) Functions Applied

As assumed by Eq.(1), \(w(i)\) integrates an effort function \(p(i)\). It is obvious that, if testing is performed well, \(w(i)\) might discover many faults. Therefore, this ease of test function is dependent on the test progress. In [9, 2, 3, 4], we have presented various functions for the ease of test function as given in Table 1. But, we lacked the real interpretation and especially did not clearly understand the notions of “effort” and “test progress”.

Parameters to be estimated are \(E[a]\), \(E[b]\), \(E[c]\), \(E[p]\), \(E[q]\), and \(E[r]\), respectively. Eq.(a) is called the Information Enhanced \(w(i)\)-function. \(I(i)\) represents test-process information such as the number of test-workers (\(tester(i)\)) involved or the number of test cases executed at a test instance \(t(i)\). Eq.(e) and Eq.(f) are the \(w(i)\)-functions that reflect the mutual relationship of the HGDM to the NHPP Goel-Okumoto model and the Delayed S-shaped model, respectively [2], [3].

In this paper we show the relationship of \(w(i)\) to the test coverage information when performing test cases. Test coverage is defined and a measure of some specific criterion; for instance the number of branches or paths taken, the number of statements executed while testing the software. Other measures could be the number of functions, modules or tasks run at test. Such test coverage information can be integrated into the estimation process of the HGDM. For describing the test coverage progress mathematically, this time we would use the linear function, such as \(w(i) = E[m] \cdot (a \cdot i + b)\). Other functions are subject to future research activities.

The relationship of \(w(i)\) to the test coverage information will be explained in the next section. The application of the HGDM, with the integration of test coverage information into the effort function, will be illustrated by real observed test-and-debug data.

3 Test Coverage Measurement and HGDM

In this section, we show the relationship of HGDM to the test coverage information collected at test-and-debugging. The test coverage information will be integrated into the ease of test function \(w(i)\).

3.1 Test Coverage Measurements

As defined in [1], test coverage is any metric of completeness with respect to a test selection criterion. Test coverage measures usually mean branch or statement coverage. Branch coverage, also denoted as \(C2\) coverage, is a metric for the number of branches executed when testing. We desire a complete coverage which means a 100% branch coverage is realized when every branch in a software has been executed at least once at testing. Statement coverage is also a metric for the number of source code lines that have been executed at test. Statement coverage is also denoted as \(C1\) coverage.

Other test coverage measures include node coverage and path coverage. These coverage notions are graph-oriented. Nodes represent the nodes in the program's control flowgraph. Links are the links in the flowgraph, path coverage looks at the execution of all possible flow paths through the program.

These coverage measures are used to certify software reliability. For achieving high reliability, 100% coverage for each of these strategies is desired. But as software testing practice shows, firstly, it is difficult in many cases to control coverage at testing, and secondly, it is hard to create test cases that might achieve 100% coverage. In many cases, software reliability is estimated without much information about the test coverage achieved at testing.

Since path coverage, branch coverage, and others are difficult to control, other simpler coverage notions
are used, such as module coverage, function coverage, task coverage. These measures are obviously much weaker in assessing reliability than those mentioned earlier. But even such retrievable information can be of greatest use to assess reliability.

In this paper, we introduce the test coverage function tc(i) as a linear equation showing the trend in the coverage while testing. In general, over a whole period of test, coverage might be increasing, constant, or even decreasing. It is rather desired that test coverage is growing as test-and-debugging proceeds. Therefore, test coverage can be represented as

\[
tc(i) = a_{tc} \times i + b_{tc}, \quad 0 \leq i \leq n. \tag{5}
\]

\(a_{tc}\) and \(b_{tc}\) can be any real as long as \(0.0 \leq tc(i) \leq 1.0\) on the interval of the real observed test-and-debug data \(n\). The test coverage measure will be expressed as a percentage of the whole coverage. We adopt a scale of \(0.0 \leq tc(i) \leq 1.0\) to represent percentage values. If \(a \leq 0.0\), then testing faces a decrease in the test coverage.

Functions other than Eq. (5) are subject to future research activities. It is obvious that more flexible \(tc(i)\) functions represent a better real behavior than a linear function.

3.2 Test Coverage Information and the HGDM

The assumptions of the HGDM state that \(w(i)\) integrates an effort function \(p(i)\). This function represents the effort needed to discover and rediscover faults at test instances \(t(i)\). We would like to show that \(p(i)\) can be related directly to \(tc(i)\), the test coverage measure. The following have been observed for \(w(i)\):

1. \(w(i)\) is limited. As indicated by Eq. (1),

\[
w(i) \leq E[m], \quad \text{with } w(i) = E[m] * p(i), \quad 0 \leq i \leq n.
\]

2. \(w(i)\) is related to the test coverage at \(t(i)\).

Test coverage information obtained during testing can be regarded as effort information.

(a) First, if the test coverage \(tc(i) = 0\), then \(w(i) = 0\), because no faults can be discovered when no software has been executed. We have

\[
.tc(i) = 0, \quad w(i) = 0. \tag{6}
\]

(b) A second case for combining test coverage information and fault detectability is more important and more complex. If \(0.0 < tc(i) \leq 1.0\), \(w(i)\) can vary in range, such as \(0.0 \leq w(i) \leq E[m]\). In this case it depends on the test case how many of \(E[m]\) faults will be discovered. Even if the test coverage measure is high, a badly chosen test case might be unable to discover many faults. Also, if \(tc(i)\) is low, but test cases are well chosen and prepared they might discover many faults.

We introduce the experience-based discovery ability parameter \(c_i\) which indicates the strength (ability) of a test case at an instance \(t(i)\) to discover faults. If test cases are more or less of the same goodness, \(c_i\) might be constant. In other cases, if test cases are of various nature, \(c_i\) is observed to change with as these test cases are applied. The following test coverage dependent case of test function is proposed:

\[
w(i) = E[m] * p(i) = E[m] * c_i * tc(i). \tag{7}
\]

With \(tc(i) = a_{tc} \times i + b_{tc}\), the new case of test function can be given as:

\[
w(i) = E[m] * c_i * (a_{tc} \times i + b_{tc}), \tag{8}
\]

for all test cases \(i = 1...n\).

(c) Third, in the special case of \(tc(i) = 1.0\), we have complete test coverage. If all of \(E[m]\) faults are discovered and rediscovered, then

\[
\begin{align*}
& \text{with } tc(i) = 1.0, \quad \text{and } w(i) = E[m] \\
& \Rightarrow c_i = 1.0. \tag{9}
\end{align*}
\]

It means that this test instance has been the most powerful.

It is important to consider the strength of test cases. These values can be experience-based from past projects, thus we can make estimation on the test progress. When estimating those values, the observation of low strength might lead us to rewrite test cases as to achieve higher test performance.

3. Decreasing fault discovery progress. With \(p(i) = E[a] \times i + E[b]\), in our past publications, negative parameter values have been calculated for some real observed data sets. This situation could not be understood well at that moment. If we take \(p(i)\) as a test coverage measure, it might be that the test coverage shows a decreasing trend as testing proceeds. In this case, the parameter \(E[a] < 0.0\).

Thus, if the estimate for \(E[m]\) is high, and the test coverage shows a decreasing behavior, we would be better off changing the test process.

The usage of Eq. (8) will be shown by application to the example of real-observed test-and-debug data of the following chapter.

4 Real Observed Data

A set of real observed data has been collected for a small inhouse application. Specifically, test coverage information has been collected at test. The test-and-debug process and as well as the real observed data collection will be explained hereafter.
4.1 Test-and-Debug Process Description

The program which we used to analyze, is written in C programming language. It has approximately 2000 lines of code and consists of about 80 modules.

We began the test after removing the compiling errors. The testing period took a little more than a month, and was divided into four phases, A, B, C, and D. The first three phases are the functional tests, and the latest phase is the structural test. The details of these phases are as follows.

Phase A: This is the very fundamental test. We created some test cases to test the fundamental function of the program. These 27 test cases are very simple and small.

Phase B: This test phase uses the combination of the test cases of the phase A. The input data is a little more complicated than the one used in phase A. There are 2 test cases.

Phase C: In this test phase, real project-dependent input data is used as input to the program system, instead of self-composed data. This test case is very complicated and large. We performed only one test case.

Phase D: This phase differs very much from the other phases. After the tests of the previous three phases, we noticed that some parts of the program have not been covered by these previous test cases. Therefore, we created 6 new test cases that would cover those parts of program not covered by previous test cases.

4.2 Data Collection Description

Much information about the test process has been collected during the test, see Table 2, for each test case: number of steps, date, number of faults, test time, and test coverage for each test case.

*Number of steps* means the lines of source code of the program that have been executed. *Date* shows the date we tested a specific test case. *Number of faults* indicates the number of faults we discovered by the test case input to the program. *Test time* means the time used to test the test case, not including correcting time. The unit is minutes. But we don’t know the test time of those test cases which didn’t lead us to find faults. But testing has continued by the next test case, until the discovery of a next fault. This means that the time of that test case is included in the time of the test case that lead to the discovery of a fault. Therefore, in the Table 2 there are some blanks in the "Test Time" column. *Test coverage* information collection is a little more complicated. During the tests, the program changed day by day, because if some faults were discovered, we corrected them immediately. Therefore, we selected the program after all tests had been finished as a criterion of the test coverage. After the test of phase A to phase D, all the previously used test cases were run again for the program at that point, and calculated the test coverage. Here, test coverage means block coverage, and block is a set of statements which are always executed if one of them is executed.

The total number of faults found during the test phases A-D was 36 bugs. Afterwards, we did not collect more coverage-related information, but we know that 6 more faults have been discovered after phase D.

4.3 Real Observed Data Set

Table 2 shows the real observed data. 36 test cases caused the discovery of 36 faults. A total of 380 test minutes have been used for a testing period of 11 days.

<table>
<thead>
<tr>
<th>Test Phase</th>
<th>T#</th>
<th>Steps</th>
<th>Date</th>
<th>Flts</th>
<th>TT</th>
<th>Test Cov.</th>
</tr>
</thead>
<tbody>
<tr>
<td>A</td>
<td>1</td>
<td>60</td>
<td>4/17</td>
<td>0</td>
<td>115</td>
<td>60.52</td>
</tr>
<tr>
<td>B</td>
<td>2</td>
<td>47</td>
<td>4/18</td>
<td>2</td>
<td>35</td>
<td>56.99</td>
</tr>
<tr>
<td>C</td>
<td>3</td>
<td>49</td>
<td>4/18</td>
<td>0</td>
<td>10</td>
<td>52.81</td>
</tr>
<tr>
<td></td>
<td>4</td>
<td>42</td>
<td>4/19</td>
<td>1</td>
<td>5</td>
<td>49.41</td>
</tr>
<tr>
<td></td>
<td>5</td>
<td>47</td>
<td>4/22</td>
<td>3</td>
<td>10</td>
<td>55.29</td>
</tr>
<tr>
<td></td>
<td>6</td>
<td>45</td>
<td>4/22</td>
<td>0</td>
<td>51</td>
<td>51.11</td>
</tr>
<tr>
<td></td>
<td>7</td>
<td>43</td>
<td>4/22</td>
<td>0</td>
<td>50</td>
<td>50.46</td>
</tr>
<tr>
<td></td>
<td>8</td>
<td>48</td>
<td>4/22</td>
<td>2</td>
<td>20</td>
<td>51.63</td>
</tr>
<tr>
<td></td>
<td>9</td>
<td>47</td>
<td>4/22</td>
<td>0</td>
<td>51</td>
<td>51.24</td>
</tr>
<tr>
<td></td>
<td>10</td>
<td>44</td>
<td>4/22</td>
<td>0</td>
<td>51</td>
<td>51.11</td>
</tr>
<tr>
<td></td>
<td>11</td>
<td>47</td>
<td>4/22</td>
<td>0</td>
<td>54</td>
<td>54.64</td>
</tr>
<tr>
<td></td>
<td>12</td>
<td>45</td>
<td>4/22</td>
<td>1</td>
<td>15</td>
<td>52.55</td>
</tr>
<tr>
<td></td>
<td>13</td>
<td>45</td>
<td>4/24</td>
<td>0</td>
<td>51</td>
<td>51.37</td>
</tr>
<tr>
<td></td>
<td>14</td>
<td>46</td>
<td>4/24</td>
<td>0</td>
<td>56</td>
<td>56.59</td>
</tr>
<tr>
<td></td>
<td>15</td>
<td>44</td>
<td>4/24</td>
<td>0</td>
<td>53</td>
<td>53.73</td>
</tr>
<tr>
<td></td>
<td>16</td>
<td>47</td>
<td>4/24</td>
<td>0</td>
<td>52</td>
<td>52.42</td>
</tr>
<tr>
<td></td>
<td>17</td>
<td>46</td>
<td>4/24</td>
<td>0</td>
<td>54</td>
<td>54.12</td>
</tr>
<tr>
<td></td>
<td>18</td>
<td>46</td>
<td>4/24</td>
<td>0</td>
<td>51</td>
<td>51.24</td>
</tr>
<tr>
<td></td>
<td>19</td>
<td>44</td>
<td>4/24</td>
<td>0</td>
<td>51</td>
<td>51.11</td>
</tr>
<tr>
<td></td>
<td>20</td>
<td>47</td>
<td>4/24</td>
<td>1</td>
<td>25</td>
<td>52.94</td>
</tr>
<tr>
<td></td>
<td>21</td>
<td>47</td>
<td>4/24</td>
<td>1</td>
<td>25</td>
<td>52.94</td>
</tr>
<tr>
<td></td>
<td>22</td>
<td>44</td>
<td>4/25</td>
<td>0</td>
<td>52</td>
<td>52.55</td>
</tr>
<tr>
<td></td>
<td>23</td>
<td>46</td>
<td>4/25</td>
<td>0</td>
<td>51</td>
<td>51.90</td>
</tr>
<tr>
<td></td>
<td>24</td>
<td>46</td>
<td>4/25</td>
<td>0</td>
<td>53</td>
<td>53.99</td>
</tr>
<tr>
<td></td>
<td>25</td>
<td>44</td>
<td>4/25</td>
<td>0</td>
<td>50</td>
<td>50.59</td>
</tr>
<tr>
<td></td>
<td>26</td>
<td>47</td>
<td>4/25</td>
<td>0</td>
<td>50</td>
<td>50.85</td>
</tr>
<tr>
<td></td>
<td>27</td>
<td>45</td>
<td>4/25</td>
<td>0</td>
<td>51</td>
<td>51.63</td>
</tr>
<tr>
<td></td>
<td>28</td>
<td>154</td>
<td>4/26</td>
<td>4</td>
<td>20</td>
<td>76.60</td>
</tr>
<tr>
<td></td>
<td>29</td>
<td>185</td>
<td>4/27</td>
<td>0</td>
<td>15</td>
<td>75.36</td>
</tr>
<tr>
<td>C</td>
<td>30</td>
<td>720</td>
<td>4/27</td>
<td>1</td>
<td>20</td>
<td>59.22</td>
</tr>
<tr>
<td></td>
<td>30</td>
<td>720</td>
<td>5/07</td>
<td>11</td>
<td>5</td>
<td>59.22</td>
</tr>
<tr>
<td></td>
<td>31</td>
<td>36</td>
<td>5/17</td>
<td>0</td>
<td>36</td>
<td>36.08</td>
</tr>
<tr>
<td></td>
<td>32</td>
<td>42</td>
<td>5/17</td>
<td>0</td>
<td>37</td>
<td>37.65</td>
</tr>
<tr>
<td></td>
<td>33</td>
<td>35</td>
<td>5/17</td>
<td>0</td>
<td>28</td>
<td>28.76</td>
</tr>
<tr>
<td></td>
<td>34</td>
<td>54</td>
<td>5/17</td>
<td>45</td>
<td>60</td>
<td>60.39</td>
</tr>
<tr>
<td></td>
<td>35</td>
<td>24</td>
<td>5/21</td>
<td>0</td>
<td>23</td>
<td>23.27</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>56</td>
<td>5/21</td>
<td>0</td>
<td>20</td>
<td>20.55</td>
</tr>
<tr>
<td></td>
<td>36</td>
<td>380</td>
<td>11</td>
<td>36</td>
<td>380</td>
<td></td>
</tr>
</tbody>
</table>

TC# = Test Case Number; Flts = Faults; TT = Test Time Used; Test Cov. = Test Coverage

Table 2: Original Observed Test-and-Debug Data

The data of Table 2 can be reinterpreted putting emphasis on the date of test and the test-time used. Here, the test coverage measure represents the coverage of all test cases performed on that specific test day. Therefore, the coverage values are different and higher in Table 3 than in Table 2.

4.4 Change of Real Observed Data Set

We have observed that the test period A has been rather long, but the test coverage did not change very much. During this testing period, 27 simple test cases have been produced to test the software package. Furthermore, this testing work only discovered 2 new faults. Looking at the test coverage growth rate, no progress can be found. Therefore, we would have been...
better off cutting the test period A after the 8th test case, and start a new test phase, for example test phase B. Therefore, the data of Table 2 will be reduced, as shown in Table 4, discarding test cases 9 - 27. Table 4 shows the real observed number of faults as well as its related test coverage information for 17 test cases. Table 5 depicts the test-and-debug data for 9 test days.

<table>
<thead>
<tr>
<th>Date</th>
<th>Time Test</th>
<th>Number of Test Cases</th>
<th>% of Test Coverage</th>
<th>Flts</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/17</td>
<td>115</td>
<td>1</td>
<td>60.32</td>
<td>6</td>
</tr>
<tr>
<td>1/18</td>
<td>45</td>
<td>2</td>
<td>61.70</td>
<td>2</td>
</tr>
<tr>
<td>1/19</td>
<td>5</td>
<td>1</td>
<td>49.41</td>
<td>1</td>
</tr>
<tr>
<td>1/22</td>
<td>45</td>
<td>8</td>
<td>64.58</td>
<td>6</td>
</tr>
<tr>
<td>1/24</td>
<td>25</td>
<td>9</td>
<td>67.32</td>
<td>1</td>
</tr>
<tr>
<td>1/25</td>
<td>10</td>
<td>6</td>
<td>64.18</td>
<td>0</td>
</tr>
<tr>
<td>1/26</td>
<td>20</td>
<td>1</td>
<td>76.60</td>
<td>4</td>
</tr>
<tr>
<td>1/27</td>
<td>35</td>
<td>1+1</td>
<td>79.35</td>
<td>1</td>
</tr>
<tr>
<td>5/07</td>
<td>5</td>
<td>(1)</td>
<td>59.22</td>
<td>11</td>
</tr>
<tr>
<td>5/17</td>
<td>55</td>
<td>4</td>
<td>66.27</td>
<td>4</td>
</tr>
<tr>
<td>5/21</td>
<td>20</td>
<td>2</td>
<td>64.44</td>
<td>0</td>
</tr>
<tr>
<td>Total</td>
<td>330</td>
<td>17</td>
<td>36</td>
<td>34</td>
</tr>
</tbody>
</table>

Table 3: Test-and-Debug Data per Date of Test

5 Estimations and Observations

We present the estimation results obtained by HGDM and compare them to those obtained by other models. Since the main focus in this paper is on the test coverage measure, estimations by HGDM are studied precisely. The following results are depicted: (1) estimations without test coverage information integration, (2) test coverage analysis and trend analysis, (3) comparison of parameter values; comparison of test coverage $tc(j)$ to HGDM's effort function $p(j)$, (4) estimations by HGDM, including test coverage information, (5) comparison of HGDM estimates to those of other models.

5.1 Estimations Based on 36 Test Cases

The real observed data of Table 2 is used for estimations and comparison. 36 test cases have discovered 36 faults.

5.1.1 HGDM Estimations Without Test Coverage Knowledge

The following estimation results have been obtained for the exponential model, the Delayed S-shaped model and HGDM. "Est. Method EF2" represents the Least Square fitting method and "Est. Method EF3" the Maximum Likelihood Method. These two methods have been applied to calculate the optimal parameter values. "Goodness of Fit EF2" gives the fitting value for $\frac{1}{n} \sum (C(i) - E(C(i)))^2$ for $i = 1...n$ test instances.

<table>
<thead>
<tr>
<th>Model</th>
<th>EF</th>
<th>$E_{[m]}$</th>
<th>$E_{[a, \varphi, \rho]}$</th>
<th>$E_{[b]}$</th>
<th>$GF$</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp.</td>
<td>EF2</td>
<td>179.65</td>
<td>0.0054</td>
<td>---</td>
<td>23.65</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>$\infty$</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
<tr>
<td>Del.S.</td>
<td>EF2</td>
<td>36.12</td>
<td>0.0880</td>
<td>---</td>
<td>34.66</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>51.500</td>
<td>0.0676</td>
<td>---</td>
<td>41.40</td>
</tr>
<tr>
<td>HGDM</td>
<td>EF2</td>
<td>179.67</td>
<td>0.0000</td>
<td>0.00539</td>
<td>23.65</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>$\infty$</td>
<td>---</td>
<td>---</td>
<td>---</td>
</tr>
</tbody>
</table>

$EF = \text{Evaluation Function}; GF = \text{Goodness-of-Fit}(EF2)$

Table 6: Estimates for 36 Test Cases

Since this data does not show a steady growth, but flattens off after the 12th test case to sharply bend to
Figure 2: Observed Curves for 36 Test Cases

A higher level after the 28th test case, a high goodness-of-fit is difficult to realize by the models. Exponential estimates fit the real observed data the best. The estimates for $E[m]$ of the Delayed S-shaped growth curve with the Least Square Method is too optimistic. The one of EF3 might be rather realistic. As proven in [9], HGDM and exponential NHPP give the same estimates for exponential growth. Therefore, both models estimate the same number of initial faults with the same fit of the estimated curve to the real observed one, $E[m] = 179.6$. Also the fit of the estimated growth curve to the real observed growth curve is the same.

From Eq. (2), the mean value function of the HGDM can explicitly be written as:

$$E[C(i)] = 179.67 \times (1.0 - \prod_{j=1}^{i} (1.0 - p(j)))$$

with $p(j) = 0.005389$, for $i=36$ test-cases.

5.1.2 Test Coverage Trend Analysis

The test coverage measures of Table 2 are plotted in Fig. 3. Since test coverage has the unit of %, we choose a scale of $0.0 \leq tc(j) \leq 1.0$, for $j = 1...36$.

For this real observed data $tc(j)$, the best fitting linear curve has been evaluated as

$$tc(j) = -0.00085509 \times j + 0.52908872 \quad (10)$$

indicating that there is a small decrease in the test coverage when looking at the whole test period of 36 test cases. Eq.(10) is represented in Fig. 3 as “Mean Cov.” “$p(j)/0.01$” means the estimate of HGDM, divided by the assumed strength parameter $c = 0.01$.

5.1.3 Estimation Results - Comparison

Looking at the HGDM effort function $p(j) = 0.005389$, which is constant for this data set for all 36 test cases,

$$p(j) = c \times tc(j). \quad (11)$$

The numerical values for test cases $j = 1$ and $j = 36$ are given in Table 7.

Table 7: Comparison of $p(j)$ and $tc(j)$

<table>
<thead>
<tr>
<th>Test Case</th>
<th>$p(j)$</th>
<th>$tc(j)$</th>
<th>$c$</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.005389</td>
<td>0.529337</td>
<td>0.010202</td>
</tr>
<tr>
<td>36</td>
<td>0.005389</td>
<td>0.498309</td>
<td>0.010202</td>
</tr>
</tbody>
</table>

The strength of test cases is growing rather slightly over 36 test cases. The experienced-based $c$ has an average of 0.010508% which indicates that the strength of the test cases applied is rather low. Which is the case, because for the test cases $i = 9...27$, no real progress has been made for fault discovery.

5.1.4 HGDM Estimation Including Test Coverage Knowledge

We would like to show the estimate for the number of initial faults of this software package by

- assuming that the experienced-based value $c = 0.010508\%$, and
- for the parameters $E[a]$ and $E[b]$ of HGDM, we take the optimal parameter values $a_t$ and $b_t$, calculated by the linear fit of Eq.(10) to the real observed test coverage.

With $c = 0.010508\%$, $E[a] = -0.00085509$, and $E[b] = 0.52908872$, $E[m] = 178.808$ initial faults are estimated. This estimate is almost the same one as the one obtained without regarding test coverage information ($E[m] = 179.6707$). Fig. 4 shows both estimated growth curves. The goodness of fit value $FE2 = 23.566680$ is almost the same as the one obtained in Table 6.
5.2 Estimations Based on 11 Test Dates

The data of Table 3 is subject to analysis in this section. Here, estimations are based on the date of test, rather than test cases. 11 test days have lead to the discovery of 36 faults.

5.2.1 HGDM Estimations Without Test Coverage Knowledge

<table>
<thead>
<tr>
<th>Model</th>
<th>EF</th>
<th>$E[m]$</th>
<th>$E[a]$</th>
<th>$E[b]$</th>
<th>GF-EF2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp.</td>
<td>EF2</td>
<td>$\infty$</td>
<td>$-$</td>
<td>$-$</td>
<td>7.18</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>$\infty$</td>
<td>$-$</td>
<td>$-$</td>
<td>7.24</td>
</tr>
<tr>
<td>Del.S.</td>
<td>EF2</td>
<td>52.93</td>
<td>0.2105</td>
<td>$-$</td>
<td>12.73</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>48.93</td>
<td>0.2378</td>
<td>$-$</td>
<td>12.64</td>
</tr>
<tr>
<td>HGDM</td>
<td>EF2</td>
<td>808.62</td>
<td>0.00013</td>
<td>0.0034</td>
<td>6.72</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>490.88</td>
<td>0.00017</td>
<td>0.0058</td>
<td>6.83</td>
</tr>
</tbody>
</table>

Table 8: Estimates for 11 Test Days

This real observed growth curve shows a constant growth, but does not indicate any tendency to level off horizontally at final test stages. Therefore, we might speculate that the testing period has not finished yet.

The NHPP exponential model fails to give proper estimates for the number of initial faults. The estimates for $E[m]$ of the Delayed S-shaped model are rather realistic. Looking at the steep increase of the real-observed growth curve, we can not exclude the estimates of the HGDM. No level-off tendency can be seen, therefore, the estimates for $E[m]$ can be accepted. Estimated growth curve of HGDM fit the real-observed data the best.

5.2.2 Test Coverage Trend Analysis

For this real observed data $tc(j)$, the best fitting linear curve has been calculated as

$$tc(j) = 0.00964818 \times j + 0.59082909 \quad (12)$$

Figure 4: Estimation with Test Coverage

5.2.3 Estimation Results - Comparison

Comparing the effort functions $p(j)$ of HGDM to the test coverage function $tc(j) = 0.00964818 \times j + 0.59082909$, we calculate $c$ from Eq.(12) by Eq.(11). The numerical values for test days $j = 1$ and $j = 11$ are given in Table 9 for each EF2 and EF3.

The experienced-based $c$ of EF3 has an average of 0.010597% which resembles very strongly to the one of the analysis method for 36 test cases. The average value for EF2, 0.0064975, is lower, which can be explained by the fact that the estimate for $E[m]$ is much higher, almost double. Therefore, the strength of test days is reduced by about half.

Figure 5: Observed Curves for 11 Test Days

Figure 6: Test Coverage for 11 Test Days
5.2.4 HGDM Estimation Including Test Coverage Knowledge

Assuming that the strength of test days \( c = 0.010597\% \) and that the parameters of HGDM are those calculated values \( a_c \) and \( b_c \) of the linear fit of Eq.(12) to the real observed test coverage ( \( E[a] = 0.00964818 \), \( E[b] = 0.59082909 \), \( E[m] = 492.877 \) initial faults. This estimate is almost the same one as the one obtained without regarding test coverage information ( \( E[m] = 490.8885 \)). Fig. 7 shows both estimated growth curves. The fit to the real observed data is almost the same with \( EF2 = 6.955672 \) this time.

<table>
<thead>
<tr>
<th>Test Day</th>
<th>( p(j) )</th>
<th>( tc(j) )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF2: 1</td>
<td>0.004565</td>
<td>0.60047727</td>
<td>0.005839</td>
</tr>
<tr>
<td>11</td>
<td>0.004876</td>
<td>0.6998907</td>
<td>0.006996</td>
</tr>
<tr>
<td>EF3: 1</td>
<td>0.006050</td>
<td>0.69987727</td>
<td>0.010075</td>
</tr>
<tr>
<td>11</td>
<td>0.007750</td>
<td>0.6998907</td>
<td>0.011119</td>
</tr>
</tbody>
</table>

Table 9: Comparison of \( p(j) \) and \( tc(j) \)

5.3 Estimations Based on 17 Test Cases

The real observed data of Table 4 is used for estimations and comparison. It has been observed that test cases \( i = 9...27 \) did not change very much in the test coverage of this software. Thus an earlier change into a new test phase is very recommendable.

5.3.1 HGDM Estimations Without Test Coverage Knowledge

This data shows an S-shaped behavior, therefore the S-shaped models perform better in this case. The best fit is realized by the Least Square Method for the HGDM Model. All estimates for \( E[m] \) are rather realistic. The one obtained for the exponential model with EF2 is a rather high, due to the steep slope of the observed growth curve. It is known that 8 bugs have been discovered after testing; a total of 42 bugs have been found.

5.3.2 Test Coverage Trend Analysis

For this real observed data \( tc(j) \), the best fitting linear curve has been evaluated as

\[
tc(j) = -0.00895098 \ast j + 0.60220588 \quad (13)
\]

indicating that there is a small decrease in the test coverage when looking at the whole test period of 17 test cases. Eq.(13) is represented in Fig. 9 as "Mean Cov.". The linear trend of the test coverage shows a
decrease. HGDM however estimates an increase, as depicted by \( p(j)/0.11, \text{EF2} \) and \( p(j)/0.20, \text{EF3} \).

5.3.3 Estimation Results - Comparison
Looking at the effort function \( p(j) = 0.002524 \times j + 0.032659 \) and \( p(j) = 0.006498 \times j + 0.040006 \), we can compare these functions to the linear test coverage growth function, \( tc(j) = -0.00085509 \times j + 0.52908872 \), deducting the experienced-based strength parameter \( c \).

<table>
<thead>
<tr>
<th>Test Case</th>
<th>( p(j) )</th>
<th>( tc(j) )</th>
<th>( c )</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF2: 1</td>
<td>0.0351383</td>
<td>0.5932549</td>
<td>0.0393050</td>
</tr>
<tr>
<td>17</td>
<td>0.075967</td>
<td>0.450392</td>
<td>0.1879120</td>
</tr>
<tr>
<td>EF3: 1</td>
<td>0.04504</td>
<td>0.5932549</td>
<td>0.076879</td>
</tr>
<tr>
<td>17</td>
<td>0.150472</td>
<td>0.450392</td>
<td>0.3343531</td>
</tr>
</tbody>
</table>

Table 11: Comparison of \( p(j) \) and \( tc(j) \)

For this data set, the strength of the test cases is growing rapidly over the 17 test cases. This indicates that a progress in the test has been made by changing earlier from test phase A to test phase B. The experienced-based \( c \) has an average of 0.1136085% for EF2, and 0.206371% for EF3.

5.3.4 HGDM Estimation Including Test Coverage Knowledge

As can be seen in Table 11, the strength of test cases \( c \) is growing as test-and-debugging proceeds. Therefore, we propose a linear function for the growth in the strength of test cases, such as

\[
\text{c}(j) = a_j * j + b_j
\]

thus defining the case of test function as

\[
\text{u}(j) = E[m] * (a_j * j + b_j) * (a_tj * j + b_tj) \quad (15)
\]

With \( E[a] = a_a = -0.0089608 \) and \( E[b] = b_c = 0.60220588 \) taken for the parameters of HGDM, the total number of initial faults has been calculated as \( E[m] = 77.67, a_c = 0.0032 \) and \( b_c = 0.0423 \). The EF3 goodness-of-fit is value is 8.007710, very similar to those obtained in Table 10. Fig. 10 shows the estimated growth curves for EF2 and EF3, and the estimated growth curve including the test coverage information.

5.4 Estimations Based on 9 Test Dates
The data of Table 5 is used for analysis in this section. Estimations are based on dates of test; 9 test days have discovered 34 faults. After this test period, further 8 faults have been found.

5.4.1 HGDM Estimations Without Test Coverage Knowledge

Figure 11 shows the real observed growth curves and the estimated growth curves. The shape of the real observed growth curve does not change very much compared to the one of 11 days of test.

Figure 10: Estimation with Test Coverage

![Figure 10](image)

Figure 11: Observed Curves for 9 Test Days
The estimates for \( E[m] \)'s of the Delayed S-shaped model and HGDM with EF3 are rather realistic. HGDM with EF2 estimates the best fit to the real observed growth curve.

<table>
<thead>
<tr>
<th>Model</th>
<th>EF</th>
<th>( E[m] )</th>
<th>( E[a] )</th>
<th>( E[b] )</th>
<th>GF-F2</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exp.</td>
<td>EF2</td>
<td>( \infty )</td>
<td>-0.0088</td>
<td>-0.0038</td>
<td>5.72</td>
</tr>
<tr>
<td>Del.S.</td>
<td>EF2</td>
<td>44.35</td>
<td>0.008</td>
<td>-</td>
<td>6.11</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>54.34</td>
<td>0.243</td>
<td>-</td>
<td>7.13</td>
</tr>
<tr>
<td>HGDM</td>
<td>EF2</td>
<td>185.80</td>
<td>0.001</td>
<td>0.0159</td>
<td>4.91</td>
</tr>
<tr>
<td></td>
<td>EF3</td>
<td>54.68</td>
<td>0.011</td>
<td>0.0460</td>
<td>5.57</td>
</tr>
</tbody>
</table>

Table 12: Estimates for 9 Test Days
5.4.2 Test Coverage Trend Analysis

For this real observed test coverage data \( tc(j) \), the best fitting linear curve has been evaluated as

\[
tc(j) = 0.010171818 \times j + 0.58506909
\]  

(16)
indicating that there is only a small increase in the test coverage looking at the whole test period of 9 test days. Eq.(16) is represented in Fig. 12 as "Mean Cov.".

HGDM also estimates an increase in the strength of test days, as depicted by "p(j)/0.036, EF2" and "p(j)/0.159, EF3".

11 TD/EF3
17 TC/EF2
17 TC/EF3
average
10^{-3}
Test Coverage
0
Mean Cov.
Cumulative Number of Faults

Figure 12: Test Coverage for 9 Test Days

5.4.3 Estimation Results - Comparison

Considering the effort functions \( p(j) \) of HGDM to the test coverage function \( tc(j) = 0.00964818 \times j + 0.59082909 \), the following strength values can be calculated, as depicted in Table 13.

<table>
<thead>
<tr>
<th>Test Day</th>
<th>p(j)</th>
<th>tc(j)</th>
<th>c</th>
</tr>
</thead>
<tbody>
<tr>
<td>EF2: 1</td>
<td>0.0174</td>
<td>0.595240908</td>
<td>0.029232</td>
</tr>
<tr>
<td>9</td>
<td>0.0294</td>
<td>0.676618452</td>
<td>0.043452</td>
</tr>
<tr>
<td>EF3: 1</td>
<td>0.0576</td>
<td>0.595240908</td>
<td>0.096768</td>
</tr>
<tr>
<td>9</td>
<td>0.1504</td>
<td>0.676618452</td>
<td>0.222283</td>
</tr>
</tbody>
</table>

Table 13: Comparison of \( p(j) \) and \( tc(j) \)

The experienced-based \( c \) of \( EF2 \) has an average of 0.036342% and the one of \( EF3 \) is 0.1595255%. \( EF2 \) estimates less strength for test days due to a high \( E[m] \) estimate. For \( EF3 \), the strength of test days is increasing rapidly.

5.4.4 HGDM Estimation Including Test Coverage Knowledge

Also in this case, Eq.(15) is used to represent the change in the strength of test days. Parameter values are the following: with \( E[a] = a_c = 0.010171818 \) and \( E[b] = b_c = 0.58506909 \), and \( E[m] = 117.8 \) initial faults have been estimated. In this case, \( a_c = 0.0040 \) and \( b_c = 0.0420 \). The \( EF3 \) goodness-of-fit is 4.910501, which is even better than the one calculated in Table 12. Fitting has increased here.

Table 14: Estimation Result Comparison

<table>
<thead>
<tr>
<th>Data Set</th>
<th>( E[m] )</th>
<th>( a_c )</th>
<th>( b_c )</th>
<th>( c : t = 1 - r )</th>
</tr>
</thead>
<tbody>
<tr>
<td>36 TC/EF2</td>
<td>179.6</td>
<td>-0.00085</td>
<td>0.5290</td>
<td>0.0102 - 0.0106</td>
</tr>
<tr>
<td>17 TC/EF2</td>
<td>178.8</td>
<td>-0.00085</td>
<td>0.5290</td>
<td>0.0100 - 0.0111</td>
</tr>
<tr>
<td>17 TC/EF3</td>
<td>490.8</td>
<td>0.00964</td>
<td>0.5908</td>
<td>0.0100 - 0.0111</td>
</tr>
<tr>
<td>17 TC/EF3</td>
<td>492.8</td>
<td>0.00964</td>
<td>0.5908</td>
<td>0.0100 - 0.0111</td>
</tr>
<tr>
<td>9 TD/EF2</td>
<td>179.6</td>
<td>-0.00085</td>
<td>0.5290</td>
<td>0.0102 - 0.0106</td>
</tr>
<tr>
<td>9 TD/EF3</td>
<td>178.8</td>
<td>-0.00085</td>
<td>0.5290</td>
<td>0.0100 - 0.0111</td>
</tr>
<tr>
<td>9 TD/EF3</td>
<td>490.8</td>
<td>0.00964</td>
<td>0.5908</td>
<td>0.0100 - 0.0111</td>
</tr>
<tr>
<td>9 TD/EF3</td>
<td>492.8</td>
<td>0.00964</td>
<td>0.5908</td>
<td>0.0100 - 0.0111</td>
</tr>
</tbody>
</table>

TC = Test Cases; TD = Test Days

1. For the first two data sets, it can be noticed, that both applications estimate a similar strength of the test cases, even if the estimates for \( E[m] \) are much different.
2. Also for the third and the forth data set, similar strength is calculated for test cases and test days. Here, the estimates for \( E[m] \) do not differ so much from each other.
3. With the introduction of information about the test coverage into HGDM, the estimates for \( E[m] \) thus obtained are well comparable to those without such information.
4. As the very slow growth in the strength of test cases and test days is estimated for the first two data sets, a change in the test strategy is very recommendable. Therefore, after changing the test strategy, as given by the third and forth data set,
a much stronger growth can be seen. In this sense, c can be used as test progress indicator.

As the examples show, test progress information, such as the test coverage measure can be well integrated into HGDM. There are two possible applications: (1) with the project-dependent experience values for the strength of test instances, HGDM is capable of making estimations for the test coverage during test-and-debugging. (2) If the test coverage measure is known, the strength of test instances can be estimated.

Having obtained the optimal parameter values by application to a set of real-observed data, we are able to manipulate those parameter values and see how to improve the test progress. For example, in case the estimated c does not satisfy our expectations, we need to change the test cases in order to achieve higher test coverage.

6 Conclusion

In past publications, we have introduced the hypergeometric distribution software reliability estimation growth model. The application to various real observed test-and-debug data showed that it is well applicable as a software reliability estimation model.

The HGDM Model includes various advantages. It is well applicable to all kind of data, since it can estimate exponentially growing as well as S-shaped growing bug curves. Therefore, we do not need to worry about which model fits best the real observed data.

Some of the conventional SRE models fit a curve described by a mathematical equation to the real-observed data, but the true interpretation of the parameters of the equation is missing. With HGDM, parameters can be explained concretely. There is a need for more sophisticated SRE models that integrate test progress information. With such information more reliable estimations might be possible. In the past, we have tried to integrate test-dependent information into HGDM, such as the number of testers involved in testing, or the number of test times used to observe failures, [8, 9].

In this paper, we showed the application of HGDM in relation to the test coverage achieved at testing. The basic theory of the model allows that test coverage information can be directly integrated into the mean value function of the model. It is easy to express the real observed test coverage as a linear function, such as \( \tau_{ij} = a + bj + c \). Taking those parameter values as the parameter values of the ease of test function \( u(t) \) of HGDM, \( E[\tau] \) estimates can be obtained that are well comparable to those calculated without prior test coverage inclusion. Furthermore, the ability or strength of test instances is an indicator of how well test is performed. Thus test progress is estimated from real-observed data.

We showed the application of the HGDM to a set of real observed test-and-debug data that specifically indicates the test coverage achieved for each test case or each test day.

The research results have been encouraging. Presently, we are investigating the application of HGDM to other data sets. Furthermore, the representation of the test coverage measure by a linear function, as described in this paper, might be too weak to reflect the real change in the test coverage. Therefore, we are looking at other more flexible functions [9].

References


