# Using Simulation for Assessing the Real Impact of Test Coverage on Defect Coverage

Lionel Briand, Dietmar Pfahl
*Fraunhofer Institute for Experimental Software Engineering (IESE)*
*Sauerwiesen 6, D-67661 Kaiserslautern, Germany*
*{briand, pfahl}@iese.fhg.de*

## Abstract

*The use of test coverage measures (e.g., block coverage) to control the software test process has become an increasingly common practice. This is justified by the assumption that higher test coverage helps achieve higher defect coverage and therefore improves software quality. In practice, data often shows that defect coverage and test coverage grow over time, as additional testing is performed. However, it is unclear whether this phenomenon of concurrent growth can be attributed to a causal dependency, or if it is coincidental, simply due to the cumulative nature of both measures. Answering such a question is important as it determines whether a given test coverage measure should be monitored for quality control and used to drive testing.*

*Although this is no general answer to the problem above, we propose a procedure to investigate whether any test coverage criterion has a genuine additional impact on defect coverage when compared to the impact of just running additional test cases. This procedure is applicable in typical testing conditions where the software is tested once, according to a given strategy, and where coverage measures are collected as well as defect data. We then test the procedure on published data and compare our results with the original findings. The study outcomes do not support the assumption of a causal dependency between test coverage and defect coverage, a result for which several plausible explanations are provided.*

**Keywords:** software test, test coverage, defect coverage, test intensity, simulation

## 1. Introduction

Testing is one of the most effort-intensive activities during software development [2]. A great deal of research is directed towards development of new, improved test methods. One way to better control testing - and thus improve test resource allocation - is to measure estimators (referred to as test coverage) of the percentage of defects detected during testing (referred to as defect coverage). A large number of test coverage measures have been proposed and studied. They range from simple measures counting the program blocks covered to data-flow based measures looking at the definition and use of variables.

Many of these test coverage measures have been investigated in terms of their subsumption relationships, the ease with which complete coverage can be achieved, and the ways they can be used to drive test case generation and selection [9] [12] [16] [22] [24]. Several additional studies reporting the application of test coverage measures to control or improve test efficiency have been published [3] [23] [28]. More importantly in the context of our research, researchers have attempted to build defect coverage models based on test coverage measures [1] [5] [6] [7] [8] [10] [14] [15] [17] [19] [21] [25] [29]. The basic assumption, regardless of the modeling strategy, is that there is some (significant) causal effect between test coverage and defect coverage.

However, since both test coverage and defect coverage increase with test intensity or time, it is not surprising that empirical data usually show a relationship. But it does not necessarily mean that additional test coverage drives the detection of new defects. The question investigated in this paper is how to test whether a given test coverage measurement, or several of them combined, are actually having a significant impact on defect coverage. It is also important that any solution be usable in typical testing conditions. The main focus of this paper is to present an easily replicable procedure that is based on simulation techniques and is designed to investigate the relationship between test coverage and defect coverage. Data coming from Horgan et al. [13] are used to exemplify the procedure and show how it can yield more precise results.

The structure of the paper is as follows. We first give a precise definition of the problems associated with using test coverage measures for controlling test effectiveness.

Then, in Section 3, we present a simulation-based procedure to test the impact of test coverage on defect coverage. Section 4 provides the results of applying our simulation-based procedure. The paper concludes with a summary of the work done and proposes directions for future research.

## 2. Problem Statement

When a relationship is observed between test coverage and defect coverage, it is commonly assumed to support the hypothesis that test coverage leads to defect coverage. However, is this assumption really capturing reality? This has important practical implications as it justifies why testing should be coverage driven, or evaluated based on coverage achievement. Therefore, it is important that test coverage measures be validated as significant defect coverage drivers.

Another, perhaps more plausible, interpretation of any empirical relationship between a test coverage measure and defect coverage is that they are both driven by more testing (referred to as test intensity). This is the typical dilemma of interpreting a relationship as causal or coincidental.

One way to approach the problem above would be to determine whether test coverage has any additional impact on defect coverage as compared to test intensity alone. In other words, this is equivalent to assess whether test coverage is still a statistically significant indicator of defect coverage, when the effect of test intensity has already been accounted for. One approach is to determine whether the combined effect of test intensity and test coverage can better explain the variations in defect coverage than test intensity alone. If this is the case, then one can conclude that evidence suggests that test coverage has a significant additional impact on defect coverage.

In cases where testing is mainly driven by test coverage, test intensity and test coverage cannot be differentiated. But in typical situations, this is not the case. In addition, defect and test coverage data are usually collected at a few discrete points in time, e.g., at the end of each testing phase such as unit, integration, or system testing. This requires an analysis approach taking into considerations these practical constraints. In the data set used in this paper, we will see that test intensity only take three possible values. Due to the design of the original study from which the data are drawn, all systems showed identical test intensity (i.e., number of test cases) at the end of each test phase, that is when coverage measurement was taken.

In order to test the significance of the impact of test coverage on defect coverage, we will therefore define a procedure that can be easily used in a context where defect and test coverage data are collected at a few discrete points in the testing process. This procedure will be based on Monte-Carlo simulation and can be easily automated.

## 3. Testing the Impact of Test Coverage

This section presents the procedure we use to test whether test coverage has a significant impact on defect coverage. We first present the rationale, relate it to the fundamentals of statistical testing, and then describe the procedure in detail. This procedure has been designed to work in a typical context where testing is not coverage driven and where test and defect coverage data are collected at a few points in time, e.g., at the end of testing activities such as unit, integration, or system testing.

### 3.1. Rationale

Using statistical testing terminology, our goal here is to test the following null hypothesis: test coverage has no additional impact on defect coverage as compared to test intensity alone. In order for us to test this hypothesis, we need to estimate what would be the "strength" of the relationship between test coverage and defect coverage assuming the null hypothesis would be true. If the strength of this relationship is measured in terms of goodness of fit or correlation (e.g., $R^2$), we need to estimate the expected $R^2$ distribution under the null hypothesis. Such a distribution can then be used for the purpose of statistical testing by comparing $R^2$ in the observed sample to the distribution, and assessing the probability of obtaining an equivalent or higher $R^2$. If this probability is small (say below 0.05 or 0.01), we can confidently reject the null hypothesis and assume that the impact of test coverage on defect coverage is plausible. Otherwise, the null hypothesis cannot be rejected and there is no supporting evidence that test coverage has any impact on defect coverage that is not already explained by test intensity.

The main issue now is to devise a method to compute the expected $R^2$ distribution under the null hypothesis. We typically have to work with a sample of projects for which we have defect and test coverage data (usually for several test coverage measures), corresponding to certain test intensity values (e.g., number of test cases, test effort), and collected at the end of various testing activities or phases. At an intuitive level, what the procedure below does is to use the sample test and defect coverage distributions to estimate the respective population distributions. Then, it uses these estimated population distributions to generate the expected $R^2$ distribution by only taking into account the test intensity information in the sample.

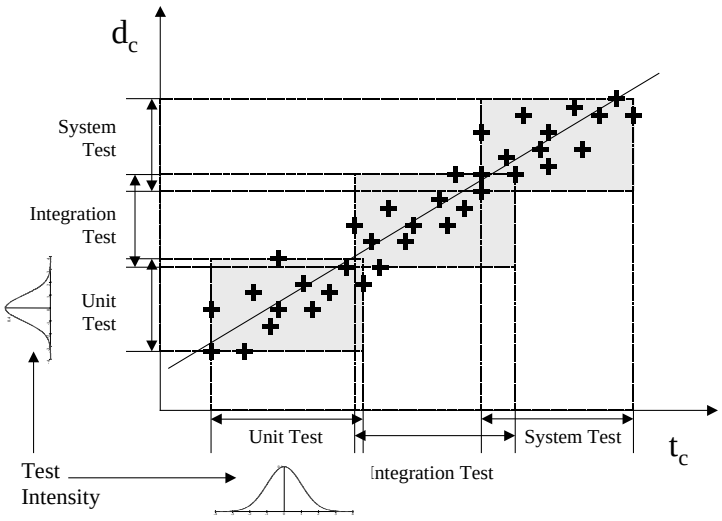


**Figure 1. Explaining the relationship between defect and test coverage.**

In order to allow for an analysis across systems of varying functional size it is recommended to normalize test intensity using any suitable measure of functional size, e.g., function points. The validity of specific test intensity and coverage measures is context-dependent and will not be discussed here, as it does not affect the procedure presented in this paper. However, it is worth noting that defining a meaningful test intensity measure requires to carefully assess their underlying assumptions. For example, if larger systems happen to be more difficult to test (diseconomies of scale), a measure of test effort per function point would not be a valid measure of test intensity. In other words, equal test intensity values may not be comparable across systems of different sizes.

In order to illustrate the principles, let us assume we only have one test coverage measure $t_c$ (e.g., block coverage) and $d_c$, capturing defect coverage. For example, although the relationship between $t_c$ and $d_c$ does not have to be linear[1], the sample data could look like the scatterplot in Figure 1. Typical test data sets are composed of data vectors containing defect and test coverage data, test effort, and number of test cases run. Across systems, such data vectors can be grouped according to test intensity levels (e.g., groups are depicted by rectangles in Figure 1). There are several strategies that can be adopted for data collection. Coverage and test intensity data can be collected at the completion of test phases (e.g., like in Figure 1), or on a regular basis, e.g., daily, weekly. The former strategy is applicable in a context of organizations with a well defined test process and strategy, which are expected to yield similar test intensities across systems for each of the test phases. Regarding the latter data collection strategy, daily measurement is probably necessary for organizations that develop small and medium software programs. For organizations that develop large software systems, over long periods of time, weekly measurement might be sufficient. In any case, it is important to collect coverage and test intensity data at sufficient granularity level to allow for the grouping, across systems, of data vectors showing similar test intensities. The number of groups does not matter so much, as long as the number of observations within groups is large enough for the estimation of the population distributions.

The example of Figure 1 shows coverage data collected from measurement of 12 programs (or any other type of object under test) and groups them according to three test phases: unit test, integration test, system test. Groups are depicted by rectangles. For the sake of simplicity, we only consider one test coverage measure here. We can imagine that, corresponding to each test phase, there are unknown population distributions for $t_c$ and $d_c$. These distributions can be then estimated using the sample distributions on these dimensions. The idea is now to generate, through simulation, a large number of scatterplots (e.g., > 1000) such as the one in Figure 1 by using the estimated population distributions on $t_c$ and $d_c$ and sampling from them independently. Each simulated sample would have exactly the same number of observations corresponding to each group as the original sample. Following that simulation procedure, the test intensity effect is preserved since we sample from the corresponding $t_c$ and $d_c$ distributions for each test phase. However, since we ignore the pairing between defect coverage and test coverage (as independent sampling is

[1] If the relationship is not linear, e.g., exponential, it can be linearized to facilitate the analysis.

performed for the two distributions), no coverage relationship is preserved if not already accounted for by test intensity. If test coverage is important, we would expect the simulated samples to show, on average, a poorer correlation than the actual sample, where both test intensity *and* test coverage effects on defect coverage should be visible. In other words, the $R^2$ distribution of the simulated scatterplots is the distribution we would expect under the null hypothesis and it can be used to test how likely is the sample $R^2$ value under this condition. Such procedure can also be used when using several test coverage measures and multivariate regression.

## 3.2 Procedure for Statistical Testing

This section presents a procedure that can be used to compute the $R^2$ distribution characterizing the relationship between $t_c$ and $d_c$ which is expected under the null hypothesis, that is when it is assumed that defect coverage is only driven by test intensity. Such a distribution can then be used for the purpose of testing whether our null hypothesis can be rejected based on available evidence, that is the actual sample $R^2$.

**3.2.1. Hypothesis definition.** The null hypothesis is defined as $H_0$: $R_{emp}^2 = R_0^2$.

$R_{emp}^2$ is the actual sample $R^2$, i.e. the (multivariate) regression coefficient $R^2$, relating $d_c$ to $t_i$ *and* $t_c$, as calculated based on the available data set.

$R_0^2$ is the (multivariate) regression coefficient under the null hypothesis ($H_0$), when defect coverage $d_c$ is related to test intensity $t_i$ only.

**3.2.2. Definitions and formalism.** The notation below is used to represent the available data on test intensity, test coverage, and defect coverage.

Test intensity: 1 to k > 1 levels of test intensity: $t_i(1)$, …, $t_i(k)$, with $t_i(1) < \ldots < t_i(k)$;

Each level may correspond to the completion of a test phase, for example. Depending on the specific context, test intensity may be measured in terms of test cases, test effort, or any other measure that adequately captures the amount of testing applied to a piece of software.

Test coverage: 1 to m > 0 test coverage measures, e.g., block, decisions; 1 to n > 0 test objects, e.g. modules, units, programs, sub-systems; A data set TC of m*n*k test coverage measurements $t_{cs}(j|h)$, with $1 \le s \le m$, $1 \le j \le n$, $1 \le h \le k$:

for $t_i(1)$: $t_{c1}(1|1)$, …, $t_{c1}(n|1)$, …, $t_{cm}(1|1)$, …, $t_{cm}(n|1)$;
for $t_i(2)$: $t_{c1}(1|2)$, …, $t_{c1}(n|2)$, …, $t_{cm}(1|2)$, …, $t_{cm}(n|2)$;
…
for $t_i(k)$: $t_{c1}(1|k)$, …, $t_{c1}(n|k)$, …, $t_{cm}(1|k)$, …, $t_{cm}(n|k)$;

Defect coverage: A data set DC of n*k defect coverage measurements $d_c(j|h)$, with $1 \le j \le n$, $1 \le h \le k$:

for $t_i(1)$: $d_c(1|1)$, …, $d_c(n|1)$;
for $t_i(2)$: $d_c(1|2)$, …, $d_c(n|2)$;
…
for $t_i(k)$: $d_c(1|k)$, …, $d_c(n|k)$;

Defect coverage is calculated as the cumulated number of defects found at a given test intensity, divided by the total number of defects.

Both test and defect coverage measurements are usually expressed as percentages.

**3.2.3. Procedure to construct the test distribution.** A five-step procedure is defined to construct the test distribution for $R^2$ under the null hypothesis (i.e., probability distribution of $R_0^2$):

*Step P1: Estimate Theoretical Distributions for all Coverage Measures*

For each coverage criterion (test and defect), find the best fit distributions $D(t_{c1}|1)$, …, $D(t_{cm}|1)$, $D(d_c|1)$ to the sample data of test intensity level $t_i(1)$. Finding the best distributions and fitting its parameters can be easily automated by using adequate tools such as BestFit [4]. Specific statistical tests (e.g., Chi-square, Kolmogorov-Smirnov, Anderson-Darling [26]) are usually helpful to find the analytical distribution, e.g., Normal, Beta, Weibull, with the closest fit to the data and to determine whether any subset is a plausible representation of the population distribution.

*Step P2: Derive Coverage Conditional Distributions for Subsequent Test Intensity Levels*

Since test coverage and defect coverage measurements are cumulated, dependencies between the distributions of coverage measurements of subsequent test intensity levels are to be expected. The dependencies between distributions $D(t_{cs}|1)$ and $D(t_{cs}|2)$, $D(t_{cs}|2)$ and $D(t_{cs}|3)$, …, $D(t_{cs}|k\text{-}1)$ and $D(t_{cs}|k)$, can be modeled by using the "envelope method" as described in [26]. If normal distributions are used, and linear dependency can be assumed, the envelope method involves running a least squares regression analyses between the same test coverage measurements of different test phases. Details on how to model dependencies between distributions by using least squares regression are given in Section 4, illustrated by an example.

To make sure that a realistic population is generated when using simulation, it is important to guarantee that dependencies within samples are explicitly modeled. The dependencies due to the cumulative nature of the data sets are to a large degree conserved by using the envelope method. However, it is still possible that simulated data sets are outside a realistic range, e.g., they may show a decreasing coverage. This is caused by the fact that fitting

based on the empirical sample may yield distributions whose domain is larger than the realistic sampling ranges. Thus, to avoid unrealistic sampling from fitted distributions, the following lower and upper bounds for test and defect coverage have to be enforced when necessary:

$0 \le t_{c1}(1) \le t_{c1}(2) \le \ldots \le t_{c1}(k) \le 100\%$, with $t_{c1}(h) \in D(t_{c1}|h)$, h = 1, …, k.

$0 \le t_{c2}(1) \le t_{c2}(2) \le \ldots \le t_{c2}(k) \le 100\%$, with $t_{c2}(h) \in D(t_{c2}|h)$, h = 1, …, k.

…

$0 \le t_{cm}(1) \le t_{cm}(2) \le \ldots \le t_{cm}(k) \le 100\%$, with $t_{cm}(h) \in D(t_{cm}|h)$, h = 1, …, k.

$0 \le d_c(1) \le d_c(2) \le \ldots \le d_c(k) \le 100\%$, with $d_c(h) \in D(d_c|h)$, h = 1, …, k.

*Step P3: Perform the Monte-Carlo Simulation*

By independently sampling from the test and defect coverage distributions modeled in Steps P1 and P2, Monte-Carlo simulation can be used to generate N data sets that conserve the distribution properties of the original data sets TC and DC. For a large N, the generated data sets should provide a representative picture of what samples should look like under the null hypothesis. This stems from the fact that defect and test coverage distributions are sampled independently for each test intensity level, thus ignoring any possible relationship between these measures. Latin Hypercube sampling [26] can be applied in order to speed up the convergence of the simulated distributions towards the theoretical population distribution from which the sample is drawn. In this context, N>1000 data sets usually provide an adequate level of precision for the estimated population distribution.

*Step P4: Derive the $R^2$ Distribution under the Null Hypothesis*

For each of the N data sets generated in Step P3, (multivariate) regression analysis is performed between defect and test coverage measures and the corresponding $R^2$ is calculated. The sample of N $R^2$'s can then be used for the purpose of statistical testing, as described below.

**3.2.4. Testing the null hypothesis.** To test the null hypothesis $H_0$, the following steps have to be performed:

*Step T1: Set $\alpha$*

Set significance level $\alpha$ to a given risk level of falsely rejecting the null hypothesis. Typical levels are 5% or 1%.

*Step T2: Calculate $R_{emp}^2$*

Calculate $R_{emp}^2$ based on data sets TC and DC by performing (multivariate) regression, e.g., by calculating the regression line $d_c = a + b_1 * t_{c1} + b_2 * t_{c2} + \ldots + b_m * t_{cm}$ in case of linear relationship. Using the notation above, $t_{cs}$ stands short for the measurement of a given coverage measure across all test intensity levels and test objects, i.e., $t_{cs}$ (*|*).

*Step T3: Perform Statistical Test*

To compare $R_{emp}^2$ with the sample of N $R^2$-values computed on simulated samples as calculated in Step P4, compute the number of $R^2$ instances that are above $R_{emp}^2$. If this number represents a percentage of N larger than $\alpha$, then the null hypothesis cannot be rejected. In this case, we cannot conclude that test coverage has a significant impact on defect coverage.

## 4. Case Study

Based on data that was generated during an experiment conducted by the University of Iowa and the Rockwell/Collins Avionics Division [13], this section illustrates how to apply the statistical testing procedure defined in Section 3.2.

### 4.1. Background Information

The purpose of the experiment was to investigate the relationship between the coverage of program constructs during testing and defect coverage. For this purpose, based on *one* specification, 12 program versions were developed independently, the program sizes ranging from 900 to 4000 uncommented lines of code. Then, test coverage and defect coverage were measured in three subsequent test phases: unit test (UT), integration test (IT), and acceptance test (AT). Because the programs were also exposed to field trials, a realistic approximation of the total number of defects (ranging from 5 to 10 defects) contained in each program could be made, thus allowing for a sensible calculation of actual defect coverage during test. An important prerequisite for the procedure defined in Section 3 is that, in each test phase, an equal level of test intensity be applied to the programs. In the experiment conducted by Horgan et al. this prerequisite was fulfilled since each program was subject to exactly the same set of test cases in each test phase. It should also be noted that information on test coverage was not used to influence testing, e.g., by driving the generation of test cases such that test coverage is systematically increased.

In the experiment, test coverage was measured for four criteria: block, decision, c-uses, and p-uses coverage [18]. Blocks and decisions are constructs contained in the control flow of a program. Typical examples of blocks in a program are consecutive code fragments that execute together and do not contain branches. Decisions are defined by the possible values of a branch predicate. C-

use and p-use are data flow oriented coverage measures. They represent special cases of definition-use pairs associated with program variables, i.e. first use of variable in a calculation, and first use of variable in a predicate after last modification (or definition). Each coverage criterion was measured for each program version at the end of each test phase using the ATAC (Automatic Test Analysis for C) tool [18].

### 4.2. Description of Available Data Set

Table 1 shows the raw data for defect coverage ($d_c$) and the four test coverage ($t_c$) measures taken from 12 test objects at three levels of test intensity (test phases UT, IT, and AT). The measurements (gray shaded area) are expressed in terms of cumulated numbers over phases, and presented in percentages.

### 4.3. Generation of Test Distribution

This section describes step by step how the procedure to construct the test distribution (Section 3.2) can be applied to the raw data presented in Table 1.

*Step P1: Estimate Theoretical Distributions for all Coverage Measures*

With the help of the tool BestFit, for each coverage criterion (test and defect), suitable analytical distributions $D(t_{block}|UT)$, $D(t_{decision}|UT)$, $D(t_{c\text{-}use}|UT)$, $D(t_{p\text{-}use}|UT)$, $D(d_c|UT)$ were derived through fitting 21 possible distributions against the sample data. The distributions of type $D(*|\{IT,AT\})$ are derived in Step P2, as they are conditional on the UT coverage values.

In all cases, and for each of the three test statistics (Chi-square, Kolmogorov-Smirnov, Anderson-Darling), the normal distribution turned out to be among the subgroup of plausible theoretical distributions. In other words, there is a high probability that the empirical data could have been produced by the fitted Normal($\mu$, $\sigma$). Based on this result, and to facilitate subsequent analysis steps (i.e., application of envelope method), we decided to use the normal distribution across the board, for all phases and criteria. Table 2 shows the fitted normal distributions for all test phases, not only UT. The main reason to do so is to make sure that assuming a normal theoretical distribution makes sense for all test phases, although the fitted distributions for IT and AT are not actually used in the simulation procedure below. Although it is expected here to be low, it is easy to check the sensitivity of the overall test results to the selected theoretical distributions by performing the analysis below, using various analytical coverage distributions.

**Table 1. Raw data (taken from [13]).**

| Test object (n = 12) | Test intensity level (k =3) | Test coverage measure (m = 4) | | | | Defect coverage measure |
|---|---|---|---|---|---|---|
| Program version | Test phase | block | c-use | decision | p-use | defect |
| 1 | UT - IT - AT | 65 - 85 - 95 | 60 - 83 - 96 | 36 - 71 - 88 | 30 - 66 - 84 | 22.22 – 66.67 – 88.89 |
| 2 | UT - IT - AT | 59 - 71 - 78 | 57 - 76 - 90 | 37 - 73 - 87 | 34 - 60 - 72 | 28.57 - 71.43 - 100 |
| 3 | UT - IT - AT | 62 - 77 - 88 | 56 - 80 - 96 | 37 - 63 - 78 | 38 - 63 - 78 | 30 - 70 - 100 |
| 4 | UT - IT - AT | 70 - 83 - 95 | 50 - 67 - 84 | 43 - 67 - 82 | 32 - 47 - 58 | 11.11 - 55.56 - 100 |
| 5 | UT - IT - AT | 44 - 74 - 88 | 45 - 70 - 87 | 27 - 60 - 77 | 26 - 58 - 74 | 42.86 - 57.14 - 100 |
| 6 | UT - IT - AT | 64 - 86 - 98 | 57 - 81 - 95 | 28 - 72 - 90 | 22 - 59 - 72 | 42.86 - 42.86 - 100 |
| 7 | UT - IT - AT | 56 - 79 - 91 | 44 - 72 - 87 | 33 - 69 - 82 | 23 - 49 - 57 | 55.56 - 88.89 - 100 |
| 8 | UT - IT - AT | 60 - 76 - 91 | 69 - 84 - 96 | 29 - 62 - 78 | 37 - 61 - 71 | 37.5 - 62.5 - 100 |
| 9 | UT - IT - AT | 68 - 80 - 90 | 56 - 74 - 86 | 42 - 63 - 79 | 42 - 59 - 72 | 22.22 - 44.44 - 100 |
| 10 | UT - IT - AT | 68 - 88 - 97 | 55 - 81 - 96 | 41 - 78 - 92 | 36 - 68 - 85 | 20 - 60 - 100 |
| 11 | UT - IT - AT | 71 - 86 - 97 | 56 - 78 - 93 | 42 - 72 - 89 | 38 - 64 - 80 | 20 - 50 - 100 |
| 12 | UT - IT - AT | 57 - 80 - 94 | 55 - 82 - 94 | 32 - 66 - 86 | 29 - 61 - 79 | 33.33 - 50 - 100 |

**Table 2. Fitted Normal distributions - Normal(Mean, Standard deviation).**

| Test intensity level | Distributions of coverage measures | | | | |
|---|---|---|---|---|---|
| | block | c-use | decision | p-use | defect |
| UT | Normal(62,7.56) | Normal(55,6.59) | Normal(35.6,5.76) | Normal(32.25,6.37) | Normal(30.52,12.53) |
| IT | Normal(80.42,5.32) | Normal(77.33,5.52) | Normal(68,5.41) | Normal(59.58,6.19) | Normal(59.96,13.07) |
| AT | Normal(91.83,5.57) | Normal(91.67,4.58) | Normal(84,5.29) | Normal(73.5,8.85) | Normal(99.07,3.21) |

*Step P2: Derive Coverage Conditional Distributions for Subsequent Test Intensity Levels (Test Phases)*

The cumulative nature of the coverage measurements contained in the sample creates dependencies between distributions of a particular coverage measure across subsequent test phases. The main dependency is caused by the monotonicity of cumulative data, e.g., for a particular program, block coverage at the end of phase IT cannot be smaller than at the end of phase UT. Figure 2 illustrates the monotonicity of a coverage measure by the fact that the related distributions shift from left to right. It can also be seen from Figure 2 that, due to overlapping distributions of subsequent phases, independent random sampling may violate the monotonicity condition. To ensure that this does not happen and that the Monte-Carlo sampling presented below is realistic, dependencies between the distributions of subsequent test phases have to be modeled explicitly through conditional distributions, e.g., IT block coverage distribution as a function of a specific UT block coverage value.

The regression results in Table 3 provide the equation of the least squares line y = a*x + b and the standard error of the y-estimate $S_{xy}$. The $S_{xy}$ statistic is the standard deviation of the vertical distances of each point from the least squares line. Least squares regression assumes that the error of the data about the least squares line is normally distributed. Thus, if y = a*x + b is the equation of the least squares line, we can model the conditional distribution as y = Normal(a*x + b, $S_{xy}$). An example is shown in Figure 3 where, for any value sampled from the fitted distribution of block coverage measurements in phase UT (block(UT)), the related conditional distribution of block coverage for IT is calculated as Normal(50.601 + 0.48089*block(UT), 4.07).

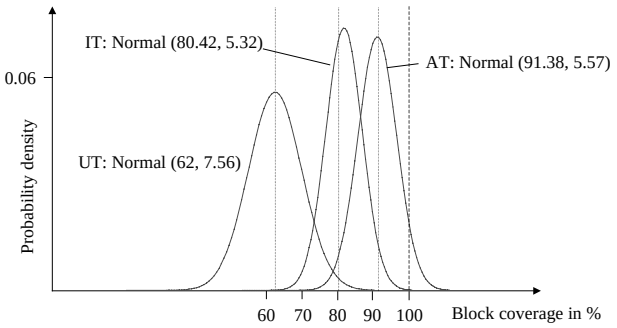


**Figure 2. Fitted distributions of block coverage measurements for subsequent test phases.**

The full set of best-fit (UT) and conditional distributions is shown in the Table 4. As expected, conditional distributions show a reduced variance. In addition to the monotonicity condition, which is taken care of by the envelope method, the table entries specify the maximum boundary (no coverage measurement can be greater than 100%) and non-negativity (no coverage measurement can be less than 0%) conditions. One special case should be noted: Since 11 out of 12 measurements of defect coverage in phase AT were equal to 100%, the envelope method was not applicable. Thus, the actual fitted distribution, with almost no variance, was used. This is clearly an idiosyncrasy of this data set as, for real-life systems, defects are likely to slip to the field.

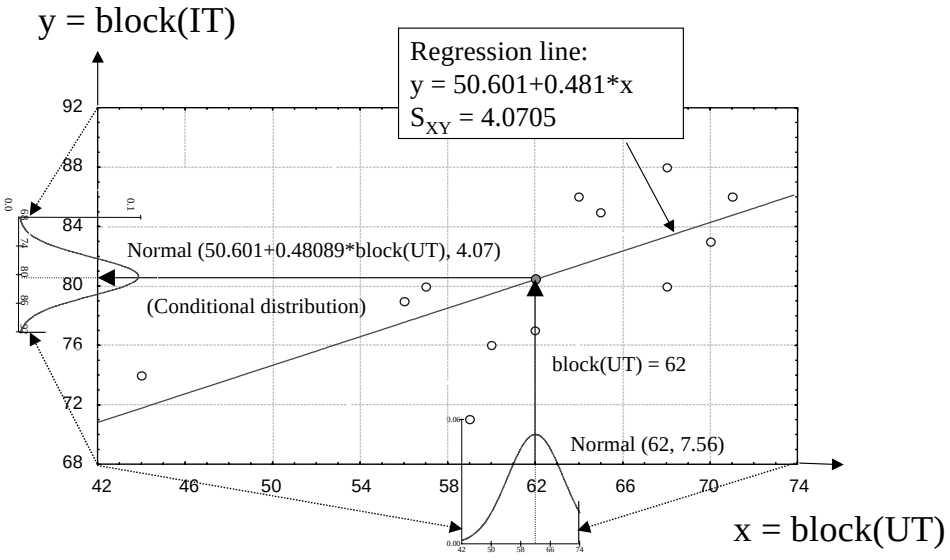


**Figure 3. Example of how to apply the envelope method.**

**Table 3. Regression summary between block coverage (UT) and block coverage (IT).**

| | Dependent Variable: IT (block)<br>R = 0.68344744, $R^2$ = 0.46710040, adjusted $R^2$ = 0.41381044<br>F(1, 10) = 8.7653, p < 0.01427, standard error of estimate: 4.0705 | | | |
|---|---|---|---|---|
| N=12 | Coefficients | Coef. Std. Error | t-value (10) | p-level |
| Intercept | a = 50.60138 | 10.13895 | 4.990792 | 0.000545 |
| UT (block) | b = 0.48089 | 0.16243 | 2.960618 | 0.014273 |

**Table 4. Summary of fitted and conditional distributions used for Monte Carlo simulation.**

| Coverage criterion | UT | IT | AT |
|---|---|---|---|
| block | **Normal (62, 7.56)**<br>Constraint:<br>if block (UT) < 0 then 0 | **Normal (50.601 + 0.481 * block (UT), 4.07)**<br>Constraints:<br>if block (IT) < block (UT) then block (UT)<br>else if block (IT) > 100 then 100 | **Normal (14.801 + 0.957 * block (IT), 2.37)**<br>Constraints:<br>if block (AT) < block (IT) then block (IT)<br>else if block (AT) > 100 then 100 |
| c-uses | **Normal (55, 6.59)**<br>Constraint:<br>if c-uses (UT) < 0 then 0 | **Normal (41.664 + 0.649 * c-uses (UT), 3.66)**<br>Constraints:<br>if c-uses (IT) < c-uses (UT) then c-uses (UT)<br>else if c-uses (IT) > 100 then 100 | **Normal (30.586 + 0.790 * c-uses (IT), 1.48)**<br>Constraints:<br>if c-uses (AT) < c-uses (IT) then c-uses (IT)<br>else if c-uses (AT) > 100 then 100 |
| decision | **Normal (35.6, 5.76)**<br>Constraint:<br>if decision (UT) < 0 then 0 | **Normal (55.909 + 0.340 * decision (UT), 5.29)**<br>Constraints:<br>if decision (IT) < decision (UT) then decision (UT)<br>else if decision (IT) > 100 then 100 | **Normal (21.913 + 0.913 * decision (IT), 1.99)**<br>Constraints:<br>if decision (AT) < decision (IT) then decision (IT)<br>else if decision (AT) > 100 then 100 |
| p-uses | **Normal 32.25, 6.37)**<br>Constraint:<br>if p-uses (UT) < 0 then 0 | **Normal (48.846 + 0.395 * p-uses (UT), 5.93)**<br>Constraints:<br>if p-uses (IT) < p-uses (UT) then p-uses (UT)<br>else if p-uses (IT) > 100 then 100 | **Normal (-8.956 + 1.384 * p-uses (IT), 2.34)**<br>Constraints:<br>if p-uses (AT) < p-uses (IT) then p-uses (IT)<br>else if p-uses (AT) > 100 then 100 |
| defect | **Normal (30.52, 12.53)**<br>Constraint:<br>if defect (UT) < 0 then 0 | **Normal (47.528 + 0.407 * defects (UT), 12.62)**<br>Constraints:<br>If defect (IT) < defect (UT) then defect (UT)<br>else if defect (IT) > 100 then 100 | **Normal (99.07, 3.21)**<br>Constraints:<br>if defect (AT) < defect (IT) then defect (IT)<br>else if defect (AT) > 100 then 100 |

**Table 5. Regression summary for multivariate linear regression with empirical sample.**

| | Dependent Variable: DEFECT<br>R = 0.88979071, $R^2$ = 0.79172750, adjusted $R^2$ = 0.76485363<br>F(4, 31) = 29.461, p < 0.00000, standard error of estimate: 14.687 | | | |
|---|---|---|---|---|
| N=36 | B | Std. Error of B | T(31) | p-level |
| Intercept | -13.3470 | 23.87424 | -0.55905 | 0.580142 |
| BLOCK | -1.1240 | 0.52737 | -2.13141 | 0.041090 |
| C_USE | 1.5858 | 0.53125 | 2.98506 | 0.005493 |
| DECISION | 1.5666 | 0.46126 | 3.39640 | 0.001889 |
| P_USE | -0.9448 | 0.50283 | -1.87887 | 0.069693 |

*Step P3: Perform the Monte-Carlo Simulation*

Using the distributions specified in Table 4, 1000 data sets were generated with Monte Carlo simulation (Latin Hypercube sampling). For each of the five coverage measures (test and defect) 36 data points were generated, 12 data points for each of the three test phases (UT, IT, AT). Simulated samples are therefore comparable to the actual sample in the sense that they are based on the same coverage distributions.

For this task we used the tools Microsoft Excel [20] and @Risk [27].

*Step P4: Derive the $R^2$ Distribution under the Null Hypothesis*

For each of the 1000 data sets generated in Step P3 the (multivariate) regression coefficient $R^2$ is calculated, that is on each data set a multivariate linear regression analysis is performed:

$$d_c = a_0 + a_1 * t_{block} + a_2 * t_{c\text{-}uses} + a_3 * t_{decision} + a_4 * t_{p\text{-}uses} + eps$$

For this task we used the tool Stata [11], which allows for an easy automation of such an iterative procedure.

The distribution of 1000 $R^2$-values, as shown in Figure 4, constitutes the distribution to be expected under the null hypothesis. As explained in the next section, this is going to be used to test whether the observed $R^2$ value is likely under the null hypothesis, i.e., test coverage has no impact of its own on defect coverage.

## 4.4. Result of Statistical Testing

Using the $R^2$ distribution described above, and to test the null hypothesis $H_0$, the following steps were performed:

*Step T1: Set $\alpha$*

The significance level $\alpha$ was set to 5%.

*Step T2: Calculate $R_{emp}^2$*

$R_{emp}^2$ was calculated based on empirical data sets TC and DC by performing multivariate linear regression[2]. The regression summary is shown in Table 5. It provides the value of the empirical regression coefficient, $R_{emp}^2 = 0.7917$, which is surprisingly high. Moreover, the low p-levels indicate that the various test coverage criteria complement each other with respect to their predictive power on defect coverage.

[2] Just by looking at the data, there was no graphical evidence of non-linearity. Of course, in case of doubt, appropriate tests for linearity and scale transformations should be performed.

*Step T3: Perform Statistical Test*

To reject the null hypothesis $H_0$, since we selected $\alpha$ = 5%, the value of $R_{emp}^2$ must be greater than the 95%-quantile of the test distribution (see Figure 4).

With $R_{emp}^2 = 0.7917 < 0.8888$ = 95%-quantile, the null hypothesis cannot be rejected. For this data set, the results do not support the claim that test coverage has a significant, additional impact on defect coverage when the effect of test intensity is already accounted for. If this data came from real projects, then nothing in the results would suggest that test case generation would be improved by following a test coverage-driven strategy.

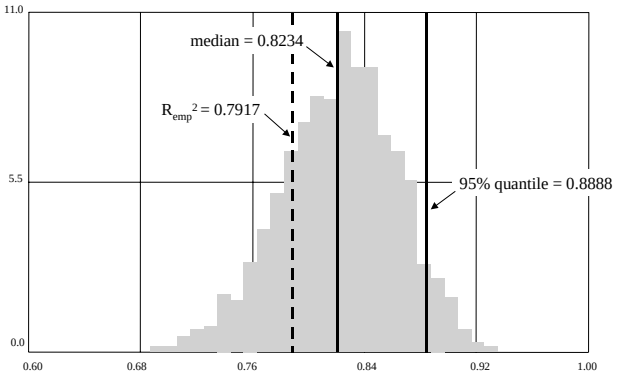


**Figure 4. Distribution of generated $R^2$-values with median and 95%-quantile.**

## 5. Conclusions and Future Work

This paper presented a simulation-based procedure to test the impact of test coverage (e.g., block coverage) on defect coverage in realistic testing and data collection conditions. The main issue raised here is that a relationship between test coverage and defect coverage does not necessarily mean there is a causal relationship. It is plausible to assume that both test coverage and defect coverage are driven by test intensity, the amount of testing performed (e.g., number of test cases). This would then lead to an empirical relationship between test and defect coverage measures. Concluding on the existence of a significant causal relationship between test coverage and defect coverage would have important practical consequences, as it would suggest that testing should be driven by such test coverage. It is therefore important to test whether test coverage really impacts defect coverage when the test intensity effect is accounted for.

We have shown here how the procedure we propose can be used and in which circumstances. The data we used did not suggest that any of the test coverage measures, even when used together, has any additional effect on defect coverage when test intensity was already accounted for. Of course, such a result is expected to vary

across environments, depending on the distribution of defects, the type of defects, etc.

This result sheds a new light on the conclusions of the original study conducted by Horgan et al., where the authors suspected that "there's a correlation between the number of faults detected in a version and the coverage of its program constructs". Based on their data, our procedure provides a precise answer to that question.

One of the preconditions of using our procedure is that test coverage is not the main driver of the testing process (e.g. the design of test cases). In cases where this is not true, then test intensity cannot really be differentiated from test coverage. This can be easily checked by looking at the relationship between the number of test cases executed and the increase in test coverage.

The relationship between defect coverage, test intensity, testability, and test coverage (or its various definitions) is complex. However, it needs to be modeled and tested in environments where early testing phases need to be controlled. In particular, we believe testability may explain the large variations observed in the relationship between defect and test coverage. Our future work encompasses the development of a series of case studies and experiments to study these complex relationships and find optimal ways to model them.

## 6. References


[1] Basili, V., Selby, R., "Comparing the Effectiveness of Software Testing Strategies", *IEEE Trans. on Software Engineering*, Vol. 13, No. 12, 1987, pp. 1278-1296.

[2] Beizer, B., *Software Testing Techniques*, Van Nostrand Reinhold, 1990.

[3] Bertolino A., Marré, M., "How many Paths are Needed for Branch Testing?", *Journal of Systems and Software*, Vol. 35, 1996, pp. 95-106.

[4] *BestFit: Probability Distribution Fitting for Windows, User's Guide*, Palisade Corporation, 1997.

[5] Del Frate, F., Garg P., Mathur, A., Pasquini, A., "On the Correlation between Code Coverage and Software Reliability", *Proc. 4th Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1995, pp. 124-132.

[6] Foreman, L., Zweben, S., "A Study of the Effectiveness of Control and Data Flow Testing Strategies", *Journal of Software and Systems*, Vol. 21, 1993, pp. 213-228.

[7] Frankl, P. G., Weiss, S. N., "An Experimental Comparison of the effectiveness of Branch Testing and Data Flow Testing", *IEEE Trans. on Software Engineering*, Vol. 19, No. 8, 1993, pp. 774-787.

[8] Frankl, P. G., Weiss, S. N., Hu, C., "All-Uses vs Mutation Testing: An Experimental Comparison of Effectiveness", *Journal of Systems and Software*, Vol. 38, 1997, pp. 235-253.

[9] Frankl, P. G., Weyuker, E. J., "An Applicable Family of Data Flow Testing Criteria", *IEEE Trans. on Software Engineering*, Vol. 14, No. 10, 1988, pp. 1483-1498.

[10] Frankl, P. G., Weyuker, E. J., "Provable Improvements on Branch Testing", *IEEE Trans. on Software Engineering*, Vol. 19, No. 10, 1993, pp. 962-975.

[11] Hamilton, L. C., *Statistics with Stata 5*, Pacific Grove, Duxbury Press, 1998.

[12] Haworth, B., "Adequacy Criteria for Object Testing", *Proceedings of the 2nd International Software Quality Week Europe*, Brussels, Belgium, Nov. 1998.

[13] Horgan, J. R., London, S., Lyu, M. R., "Achieving Software Quality with Testing Coverage Measures", *IEEE Computer*, September, 1994, pp. 60-69.

[14] Hutchins, M., Foster, H., Goradia, T., Ostrand, T., "Experiments on the Effectiveness of Dataflow- and Controlflow-Based Test Adequacy Criteria", *Proc. 16th Int'l Conf. on Software Engineering (ICSE)*, 1994, pp. 191-200.

[15] Jacoby, R., Masuzawa, K., "Test Coverage Dependent Software Reliability Estimation by the HGD Model", *Proc. 1st Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1992, pp. 193-204.

[16] Jin, Z., Offutt, J., "Coupling-based Criteria for Integration Testing, Software Testing", *Verification and Reliability*, Vol. 8, 1998, pp. 133-154.

[17] Li, N., Malaiya, Y. K., Denton J., "Estimating the Number of Defects: A Simple and Intuitive Approach", *Proc. 7th Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1998, pp. 307-315.

[18] Lyu, M. R., Horgan, J. R., London S., "A Coverage Analysis Tool for the Effectiveness of Software Testing", *Proc. 2nd Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1993, pp. 25-34.

[19] Malaiya, Y. K., Li, N., Bieman, J., Karcich, R., Skibbe, B., "The Relationship Between Test Coverage and Reliability", *Proc. 3rd Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1994, pp. 186-195.

[20] *Microsoft Excel 97, User's Guide*, Microsoft Corporation, 1997.

[21] Morgan, J. A., Knafl, G. J., "Residual Fault Density Prediction using Regression Methods", *Proc. 5th Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1996, pp. 87-92.

[22] Ntafos, S. C., "A Comparison of Some Structural Testing Strategies", *IEEE Trans. on Software Engineering*, Vol. 14, No. 6, 1988, pp. 868-874.

[23] Piwowarski, P., Ohba, M., Caruso, J., "Coverage Measurement Experience During Function Test", *Proc. 15th Int'l Conf. on Software Engineering (ICSE)*, 1993, pp. 287-301.

[24] Rapps, S., Weyuker, E. J., "Selecting Software Test Data Using Data Flow Information", *IEEE Trans. on Software Engineering*, 1985, Vol. 11, No. 4, pp. 367-375.

[25] Veevers, A., Marshall, A. C., "A Relationship Between Software Coverage Metrics and Reliability", *Software Testing, Verification and Reliability*, Vol. 4, 1994, pp. 3-8.

[26] Vose, D., *Quantitative Risk Analysis: A Guide to Monte Carlo Simulation Modelling*, John Wiley & Sons, 1996.

[27] Wayne, L. W., *Simulation Modeling Using @Risk*, Duxbury Press, 1996.

[28] Weyuker, E. J., "More Experience with Data Flow Testing", *IEEE Trans. on Software Engineering*, Vol. 19, No. 9, 1985, pp. 912-919.

[29] Wong, W. E., Horgan, J. R., London, S., Mathur, "A. P., Effect of Test Set Size and Block Coverage on the Fault Detection Effectiveness", *Proc. 3rd Int'l Symposium on Software Reliability Engineering (ISSRE)*, 1994, pp. 230-238.