Validation of Software Testing Experiments

A Meta-Analysis of ICST 2013

Mark Hays, Jane Huffman Hayes
Computer Science Department
University of Kentucky
Lexington, Kentucky, USA
mark.hays@ uky.edu, hayes@cs.uky.edu

Arne C. Bathke
Statistics Department, University of Kentucky
Lexington, Kentucky, USA; arne@uky.edu
Fachbereich Mathematik, Salzburg University
Salzburg, Austria

***Abstract*— Researchers in software testing are often faced with the following problem of empirical validation: does a new testing technique actually help analysts find more faults than some baseline method? Researchers evaluate their contribution using statistics to refute the null hypothesis that their technique is no better at finding faults than the state of the art. The decision as to which statistical methods are appropriate is best left to an expert statistician, but the reality is that software testing researchers often don't have this luxury. We developed an algorithm, MeansTest, to help automate some aspects of statistical analysis. We implemented MeansTest in the statistical software environment R, encouraging reuse and decreasing the need to write and test statistical analysis code. Our experiment showed that MeansTest has significantly higher F-measures than several other common hypothesis tests. We applied MeansTest to systematically validate the work presented at the 2013 IEEE Sixth International Conference on Software Testing, Verification, and Validation (ICST'13). We found six papers that potentially misstated the significance of their results. MeansTest provides a free and easy-to-use possibility for researchers to check whether their chosen statistical methods and the results obtained are plausible. It is available for download at coest.org.**

Keywords—software testing; statistical analysis; empirical validation

I. INTRODUCTION

Research in software testing often lends itself to empirical validation; researchers show that some new way of discovering faults finds more faults or takes less time than the state of the art. Publications often describe such results as being *significant*. While significance is an overloaded term, in the context of empirical validation, we'd like to think that significance refers to *statistical significance*: that a difference between two means or variances is not likely due to chance. Significance in this context is demonstrated through formal hypothesis testing.

Researchers who ascribe to this notion of significance must also contend with threats to *statistical conclusion validity*. Researchers sometimes incorrectly fail to reject the null hypothesis due to choosing inappropriate statistical test. Worse, they almost never go back and analyze the *statistical power* of their experiment to determine whether their sample size was appropriate. Less often, they interpret a p-value as a probability when in fact the assumptions of the underlying statistical test have not been met. While the former problem can be solved by using powerful tests, powerful tests make assumptions that run the risk of suffering the latter problem. This impasse has led researchers in software testing to rely on classical, possibly rather conservative tests from the Wilcoxon family because they "do not wish to make assumptions on the distribution." [1] We present relatively recent advances in nonparametric statistics that provide powerful alternatives to the classical tests.

In this paper, we make several contributions to address these issues. We introduce an algorithm, called MeansTest, to introduce the software testing research world to new statistical tests such as Brunner-Munzel. This process has the potential to automatically analyze experimental results. To our knowledge, there is no other approach readily available that goes through the workflow to analyze the data under study for properties such as normality, equal variance, and power, and then use that information to decide which test to apply, and then apply that test. We have encapsulated those aspects of a professional statistician's approach that can reasonably be automated; while it is obvious that common sense and experience of a trained statistician can't be replaced by an automated procedure, we also found that *some* crucial steps can indeed be left to software. The simulation of our algorithm demonstrates its usefulness. Nevertheless, the idea of automated test selection will always have an air of controversy and we recommend to the user to employ our automatism wisely, and with common sense. However, readers shall keep in mind that the only viable alternatives would be relying on unrealistic model assumptions, or restricting oneself to overly conservative tests and thus failing to detect important effects. We introduce a unique way to validate the statistical technique that blends statistics with classification-based validation seen in fields such as software fault classification. We use our statistical technique to examine prior art at the 2013 IEEE Sixth International Conference on Software Testing, Verification, and Validation (ICST'13) [2] for statistical significance.

This paper is organized as follows. Section II examines related work in empirical validation. Section III discusses the challenges that researchers face in performing valid statistical analysis. In Sections IV and V, we analyze the effectiveness of MeansTest. Sections VI and VIII state the results of our meta-analysis of ICST'13. In Section VIII, we discuss our planned future work.

978-0-7695-5185-2/14 $31.00 © 2014 IEEE
DOI 10.1109/ICST.2014.46

II. Related Work

This section presents background information for statistical analysis for empirically evaluating software testing research.

A. Definitions

To evaluate a hypothesis, researchers in software testing typically look at one or more *summary statistics*. The most common statistic is the *mean*, or average. Researchers will try to show that one approach can find more faults on average than another approach.

According to Cohen, *effect size* is a key measure in computing an experiment's *power*, which is its ability to correctly reject the null hypothesis [3]. Often, researchers will run small, initial experiments to see if their ideas hold any merit. While they may find a positive result, their result might not be statistically significant. *Post-hoc power analysis* is a procedure used at the end of such a pilot study to determine the sample size required to achieve a statistically significant result. The analysis assumes that the effect size the researcher observed will remain constant in the larger experiment.

In this work, we refer to several hypothesis tests for comparing means. Well-known hypothesis tests include the *t-tests* [4] and the *Wilcoxon* tests [5]. Less well known is the recent *Brunner-Munzel* test [6]. The t-tests are considered *parametric tests* because they assume parametrized families of distributions (normal distribution family or others where a particular distribution can typically be described using one or two parameters). The Wilcoxon and Brunner-Munzel tests are considered *nonparametric* because they do not assume that the data originates from any particular distribution family.

In particular, the t-tests assume *normality*: the assumption that the data fits a normal distribution, also known as a "Gaussian distribution" or "bell-shaped curve." The normal distribution is defined to be *symmetric*, meaning that the mean and median are equal. Symmetry does not always hold; distributions like the beta distribution can be *asymmetric*, which indicates a significant departure from normality. Distributions can also have varying *peakedness* ranging from the completely flat uniform distribution to the especially sharp t-distribution. Pearson [7] formally defined these aspects of non-normality in terms of *skewness* (asymmetry) and *kurtosis* (peakedness). Many methods of varying effectiveness exist for verifying the normality of data, but we use the *Shapiro-Wilk* test [8] because it formally reasons about the likelihood that a given sample's skewness and kurtosis are a significant departure from normality.

Some tests assume that the data takes the form of *two independent samples*. One might imagine a software testing experiment where testers are divided into two groups (control and treatment group) and testers in the treatment group are given a new method of detecting faults. We might then formulate a hypothesis that testers in the treatment group find more faults than testers in the control group; this would be an example of an experiment with two independent samples. Other software testing experiments take the form of *paired sample* experiments. In these experiments, each defect detection algorithm under study is applied to every program under test. The defects found are related (paired) on the program being tested. The experiment designer establishes whether their experiment is paired or two-sample based on the nature of the procedure; it cannot be inferred from the samples. The choice of two independent samples *vs.* paired samples determines which hypothesis tests can be applied.

Some independent samples tests, like the classical Student's t-test, also assume that the samples have *equal variance*. Several formal hypothesis tests exist for disproving this assumption, such as the classical F-test and the more robust Brown-Forsythe test [9]. However, this F-test itself makes strong assumptions about normality, so in cases where normality is not clearly inferred from the sample data, Brown-Forsythe can be more appropriate. When the data does not have equal variance, Welch's t-test is appropriate because it computes a more robust estimate of the variance.

Finally, it is our experience that the Brunner-Munzel implementation in R [10] uniquely assumes that there is *overlap* between the samples. If one were to plot two independent samples on a shared number line, the number line would indicate overlap if at least one value from one sample fell within the min and max values of the other sample. When no overlap is present, our experience is that the Brunner-Munzel test is undefined. Overlap is likely a new assumption to researchers, but as the description indicates, overlap is straightforward to test.

B. MeansTest Algorithm

In our prior work [11], we introduced the first iteration of the MeansTest algorithm. We implemented it for the experiment design framework, TraceLab [12]. MeansTest automated important aspects of the basic logic that expert statisticians use when selecting statistics tests that compare location parameters, such as the mean and median. MeansTest implemented the underlying statistical tests and testing of assumptions by invoking the statistics environment R. MeansTest reported the p-value, test statistic, and the logical path it took through the composite component. Our paper separately gave statistician-crafted language describing the meaning of each possible path so that researchers could correctly report the MeansTest output.

Figure 1 displays the precedence graph of the revised MeansTest algorithm used in this paper. MeansTest operates in two modes: paired-sample (aka "one-sample") and two-sample. In the paired mode, depending on the properties of the data being compared, MeansTest performs either the paired-sample t-test or Wilcoxon signed-rank test. In the two-sample mode, MeansTest performs one of: a) the t-test, b) Welch's t-test for unequal variances, c) Brunner-Munzel, or d) the Mann-Whitney U test (also known as the Wilcoxon rank-sum test), again depending on the shape and overlap of the two distributions.

Since its introduction, we have continued to improve to the MeansTest algorithm. We introduced post-hoc power analysis to facilitate pilot studies. In this context, power analysis takes experiment results that are not statistically significant and determines the minimum sample size required to make them significant; this analysis assumes that the observed results are

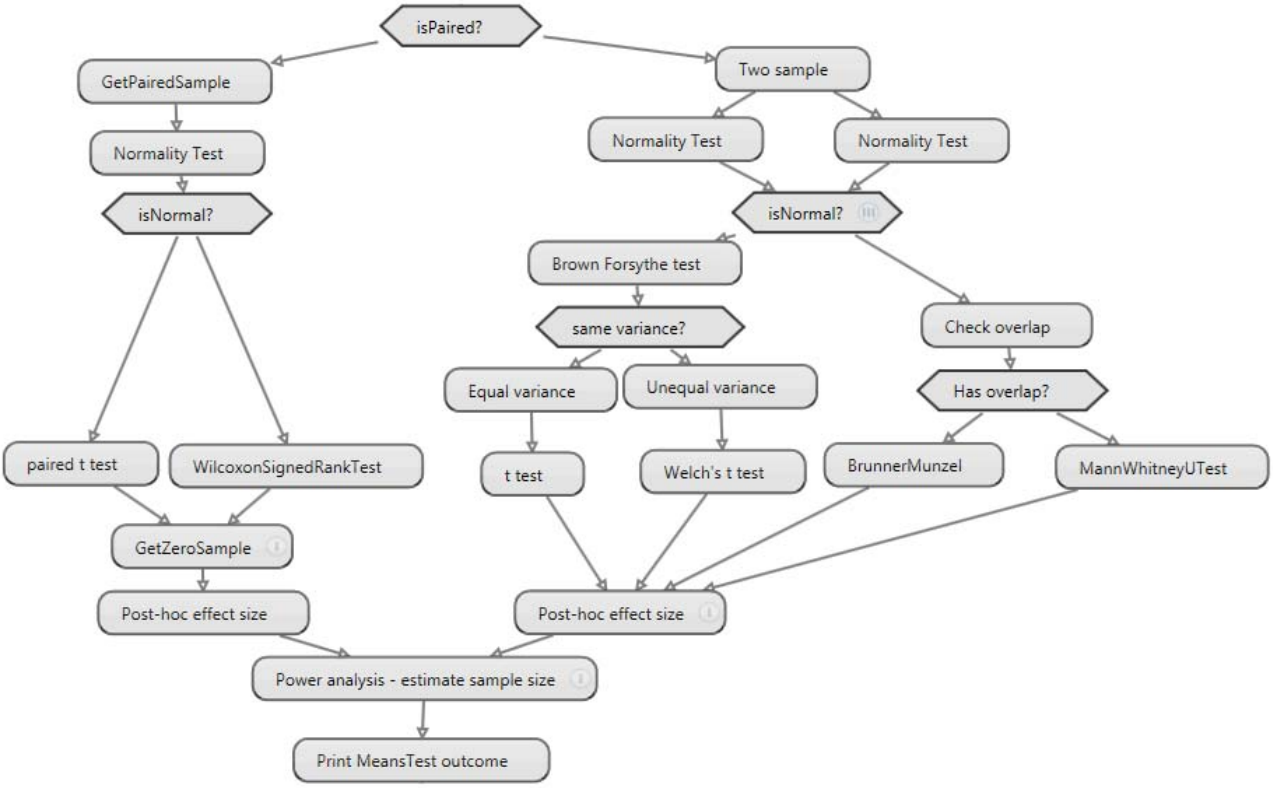

Fig. 1. The MeansTest algorithm, as implemented in TraceLab.

not an artifact of chance. Also, we now directly display in line with the results the descriptive text that a researcher can cite.

C. Tests for Normality

Razali and Wah [13] compared several algorithms for determining normality of data. Razali and Wah selected the four "most common" automated tests for normality: Shapiro-Wilk, Kolmogorov-Smirnov, Anderson-Darling, and Lilliefors. They applied these tests to samples of various sizes drawn from 14 probability distributions; they selected these distributions "to cover various standardized skewness and kurtosis values." They generated samples of sizes 10, 20, 30, 50, 100, 200, 300, 400, 500, 1000, and 2000. They found that the Shapiro-Wilk test was the most efficient overall at identifying non-normality.

The Razali and Wah paper is relevant to MeansTest for two reasons. First, they identified the best algorithm to determine non-normality; note that MeansTest used Shapiro-Wilk in the initial release. Second, they presented an empirical framework and data set that could be extended to validate MeansTest. Table 1 summarizes the probability distributions that Razali and Wah used. As Table 1 shows, the data is balanced evenly between symmetric and asymmetric distributions. It also has balanced skewness (seven values are 0) and kurtosis (seven absolute values are less than or equal to 1). This balance helps eliminate bias threats.

TABLE I. RAZALI AND WAH DISTRIBUTIONS

Distribution	Skewness	Kurtosis
uniform(0,1)	0	-1.2
beta(2,1)	-0.5656854249	-0.6
beta(2,2)	0	-0.8571428571
beta(3,2)	-0.2857142857	-0.6428571429
beta(6,2)	-0.692820323	0.1090909091
t(7,0)	0	2
t(5,0)	0	6
t(10,0)	0	1
t(300,0)	0	0.0202702703
Laplace(0,1)	0	3
chisq(4,0)	1.4142135624	3
chisq(20,0)	0.632455532	0.6
Gamma(1,5)	2	6
Gamma(4,5)	1	1.5

D. Empirical Software Engineering

Dit et al. [14] surveyed software maintenance papers from the last ten years. Their goal was to reproduce the tools and results of these papers in the TraceLab framework, then make the components publicly available. They organized the papers using the Petersen et al. systematic mapping process [15]. Their steps include: 1) Defining the research questions; 2) Conducting the search; 3) Selecting screening criteria; 4) Classifying the technique (in their case: determining the tracers

and preprocessors used); and 5) Extracting the data. We reuse this mapping process in our research.

Basili et al. surveyed empirical software engineering papers. They created a framework that describes experiments in terms of the definition, design, implementation, and interpretation of an experiment. Each part of the framework has several attributes, such as scope (single project, replicated project, multi-project variation, blocked subject-project), perspective, and impact. The authors used this framework to systematically survey papers. They identified several problem areas: 1) that there is no consistency among practitioners, 2) that experiments are hard to precisely define because there is no standard metric for gauging software quality, 3) the experiment plan should contain ideas for subsequent experiments, 4) experiments need to be published in a repeatable and extendable way, and 5) results need to be qualified by the controlled variables [16].

Kampenes et al. [17] systematically reviewed the statistical significance of 103 controlled software engineering experiments dating between 1993-2002. Their goal was to examine the proportion of journal papers that report Cohen's effect size. As a secondary measure, they collected the number of statistically significant results as reported by the authors. The 103 papers contained a total of 429 hypothesis tests. Of those, only **212 tests (49%)** indicated statistically significant results. This proportion serves as a baseline for our meta-analysis of ICST'13.

III. The Importance of Statistical Analysis

Generally, statistical inference aims to quantify whether observed real-data phenomena could be explained by chance alone, or whether the observed data are so unusual that a convincing explanation requires a model with components beyond chance alone [18]. In the case of comparing two techniques, each technique results in a data sample. Most likely, the samples will differ from each other. However, even if both techniques are equivalent, one would expect the samples to be different due to chance variation. Statistical inference provides the tools to decide whether the two samples are different enough to reject the possibility that both methods were equally effective. To this end, statistical testing procedures yield p-values. A p-value in this context is the probability, assuming both methods are indeed equivalent, that two resulting data samples would be as (or more) different than the two samples that were observed in the experiment. P-values are one of the main decision tools in statistical inference. If the p-value is small, typically less than 0.05, researchers conclude that the assumption "both methods are indeed equivalent" can no longer be upheld.

One of the main problems with p-values is that the underlying probability calculation typically relies on several model assumptions. Unless these assumptions are carefully checked and verified, the seemingly precise "p-value" can be worthless and misleading. For example, the unpaired two-sample t-test can be used to compare two independent samples. Observations in each sample are assumed to be normally distributed with equal variance. If the samples are truly independent and the observations truly follow normal distributions and have the same variance, then a reported p-value of 0.03 can indeed be interpreted as a rejection of the hypothesis "both methods are equally effective," while a p-value of 0.12 does not provide evidence against this hypothesis. However, if the samples are actually paired instead of independent, this test will often provide large p-values even if both methods are rather different. The same can happen if the data is highly skewed and thus violates the normality assumption. On the other hand, it can also happen that an inappropriately chosen statistical test provides a small p-value even though the methods being compared are not distinguishable in quality, and the observed differences are in fact due to chance. Such a test is as undesirable as the first one. In either case, the resulting p-values do not serve as a meaningful decision tool, and they do not have the probability interpretation mentioned above.

The solution to this problem is rather straightforward: only appropriate inference procedures should be used. The MeansTest workflow facilitates this by making sure that all of the important assumptions are being examined and that appropriate inference procedures are being chosen. As a result, the final reported p-value for the comparison of two methods still satisfies the probability interpretation given above. Also, this p-value can be used to decide whether the hypothesis "both techniques are equally effective" shall or shall not be rejected.

IV. Experiment Design

In our experiment, we evaluated the accuracy of the MeansTest component. We drew samples at random from the Razali and Wah probability distributions. We drew second samples at varying distances from the original samples. We then applied hypothesis tests to see if they could notice the true difference in the population means.

A. Research question

As stated earlier, the MeansTest workflow combines several hypothesis tests that researchers already use. In light of the tendency of researchers to favor the Wilcoxon tests, we might ponder whether MeansTest is more effective overall than the Wilcoxon tests. We pose this hypothesis more generally in the form of RQ0 below.

RQ0: can MeansTest more accurately detect the true significance of differences/lack thereof than other hypothesis tests?

B. Data

We studied the Razali and Wah probability distributions from Table I.

C. Procedure

We expanded the Razali and Wah procedure to cover statistical significance. Razali and Wah only evaluated the likelihood that a test of normality could find a known difference in non-normality. We looked at two aspects: the likelihood that a hypothesis test would find a significant difference *when a difference was present*, and the likelihood that a hypothesis test would not find a significant difference *when no difference was present*. This procedure is more in line with the usual fault classification experiments in software testing and gives direct evidence toward answering our research question.

We applied the following hypothesis tests: MeansTest, Student's t-test, Welch's t-test, Wilcoxon ranked-sum, and Brunner-Munzel. As mentioned earlier, these tests are all invoked by MeansTest, so it is worth considering whether MeansTest can perform any better than its parts.

We ran our hypothesis tests on many pairs of samples. Each pair consisted of an initial sample and a shifted second sample. We drew our initial samples from the Razali and Wah distributions using their sample sizes. To draw the second samples, we used Cohen's effect size to define the difficulty of noticing a difference between two samples. We systematically examined 20 effect sizes: ten of the effect sizes were selected from zero to one in 0.1 increments, while the other ten had zero effect size. This selection created a balance between zero differences and non-zero differences, reducing bias. We selected the parameters of the second sample's probability distribution to yield the desired effect size. For each distribution/sample size/effect size triple, we drew 100 pairs of initial samples and shifted samples.

The result of running each hypothesis test on each pair of samples was a p-value. We interpreted the p-value at the 95% confidence level to determine whether an outcome was significant (positive) or not significant (negative). We interpreted the correctness of this result depending on the effect size. Table 2 concisely demonstrates our classification logic given a p-value p and an effect size d.

TABLE II. CLASSIFICATION LOGIC

	Positive	Negative
True	p<0.05,d>0	p>0.05,d=0
False	p<0.05,d=0	p>0.05,d>0

We labeled these quantities using *TP* for True Positive, *TN* for True Negative, *FP* for False Positive, and *FN* for False Negative. Using these labels, we then computed the F-measure of each classification. We use the following definitions to compute F-measure [19]:

$$\text{recall} = TP/(TP + FN) \quad (1)$$

$$\text{precision} = TP/(TP + FP) \quad (2)$$

$$F = 2 * \frac{\text{recall} * \text{precision}}{\text{recall} + \text{precision}}. \quad (3)$$

The F-measure of all hypothesis tests increased with sample size, but the relative rankings between methods remained stable. To eliminate the effect of sample size on F-measure, we ordinally ranked each value at each sample size; the ranks were consistent across sample sizes. This consistency allowed us to summarize our results by distribution.

D. Hypothesis

We wanted to show that MeansTest had **higher rank** than that of other hypothesis tests. Given a hypothesis test i, Equations 4 and 5 formally state the null and alternate hypotheses as:

$$H_0: \text{Rank}(\text{MeansTest}) = \text{Rank}(i) \quad (4)$$

$$H_A: \text{Rank}(\text{MeansTest}) > \text{Rank}(i). \quad (5)$$

We rejected each null hypothesis with 95% confidence.

V. RESULTS

In this section, we describe our results, including the rankings by distribution, the p-values for the hypothesis tests, and the threats to validity.

A. Rankings

Table 3 shows the summary of the ranks on an ordinal scale of 1-5, using the average for ties. **Higher** ranks are better.

TABLE III. HYPOTHESIS TEST RANKINGS

Distribution	Means-Test	Wilcoxon	t	Welch t	Brunner-Munzel
beta(2,1)	5	3	1.5	1.5	4
beta(2,2)	3	1	4	5	2
beta(3,2)	3	1	5	4	2
beta(6,2)	5	3	2	1	4
chisq(20,0)	4	3	2	1	5
chisq(4,0)	3.5	3.5	2	1	5
Gamma(1,5)	4.5	3	2	1	4.5
Gamma(4,5)	5	2	3	1	4
Laplace(0,1)	3	5	2	1	4
t(10,0)	4	3	2	1	5
t(300,0)	4	1	5	3	2
t(5,0)	3	4	2	1	5
t(7,0)	3	4	2	1	5
uniform(0,1)	3	1	5	4	2

As Table 3 shows, MeansTest demonstrated favorable classification behavior over the other tests. While the individual tests each had their strong and weak points, MeansTest was able to infer the appropriate test sufficiently to **always place at least third or higher**. Contrast that with the Wilcoxon rank-sum test and Welch's t-test, which both fared poorly with alarming frequency. Brunner-Munzel, the new non-parametric test, usually did well. With regard to the added entry on normality, the t-tests nicely complemented Brunner-Munzel's weak points, lending credibility to our idea to pick between the tests based on the normality of the data.

B. Summary statistics

Table 4, in turn, provides summary statistics for Table 3.

TABLE IV. HYPOTHESIS TEST SUMMARY STATISTICS

Test	Worst rank	Best rank	Mean	p-value
MeansTest	**3**	5	3.8	-
Wilcoxon	1	5	**2.7**	**0.013**
Welch's t	1	5	**1.9**	**0.0037**
t	1.5	5	**2.8**	**0.038**
Brunner-Munzel	2	5	3.8	0.61

To evaluate our set of hypotheses regarding the ranks, we applied paired hypothesis testing to account for the fact that the ranks are dependent variables on the distribution being tested. Since we are here comparing ranks, only a nonparametric rank test is appropriate. Note that ranks have the property that their

sum is always constant (consider the row sums in Table 3) – therefore violating an always implicitly assumed independence assumption in parametric tests.

Using this information, we applied the Wilcoxon signed-rank test (as Brunner-Munzel only applies to independent samples). As Table 4 highlights, we found that MeansTest had a **significantly higher** rank than Wilcoxon rank-sum and the t-tests. MeansTest did not have a significantly higher rank than Brunner-Munzel.

C. Threats to Validity

In terms of threats to *statistical conclusion validity*, our results only have 95% confidence. We declared our hypotheses ahead of time so we would not have to worry about inflated experiment-wide error from performing multiple comparisons. Even if we had decided on all four tests after performing the experiment, we should still have at least 80% experiment-wide confidence according to the very conservative Bonferroni correction.

In terms of threats to *construct validity*, these should be minimal because we used the statistics framework R to perform all of our statistics. In our previous work [11], we tested each MeansTest path and confirmed that it returned the same result as the expert statisticians using other statistics software. Similarly, in our experiment, we used R to generate the probability distributions and shift their parameters.

In terms of threats to *internal validity*, we used a data set that was previously used in an experiment to assess the ability of statistics to infer non-normality, which on the surface could appear to create a bias towards nonparametric statistics. We defused this threat by establishing that the distributions were balanced between normal and non-normal skewness and kurtosis. Many of the distributions used in the experiment, such as the t distribution, were in fact approximately normal. We ran each hypothesis test on every data set we generated, so it is not possible that MeansTest received "easier" samples than the other tests.

In terms of threats to *external validity*, we only looked at the 14 Razali and Wah distributions. While we generated many samples from these distributions, it is true that there are other distributions out there such as the negative exponential and standard normal distributions. This threat is mitigated by Razali and Wah's methodology, in that they selected distributions to cover a range of standard skewness and kurtosis values. Thus, the skewness and kurtosis of many other distributions are implicitly covered by these 14 distributions.

VI. META-ANALYSIS

Borrowing from the Dit et al. mapping process, we systematically mapped the proceedings of ICST'13 into our experiment framework. We applied MeansTest to the experiments from those papers that 1) featured empirical comparisons of two or more testing methods/tools, and 2) had sufficient data in the paper to perform the validation. Based on the output from MeansTest, we reported the statistical significance of the experiments' results and provided recommendations for insignificant results.

A. Research Questions

We are curious to know:

RQ1: How pervasive were Wilcoxon tests at ICST'13?

RQ2: To what extent are the results at ICST'13 statistically significant?

B. Conducting the search

We systematically examined prior art from ICST'13. As we will show, ICST'13 was of interest because the venue featured considerable empirical validation using the Wilcoxon family of tests. ICST'13 also had several empirical validation papers which did not comment on the statistical significance of their results.

C. Screening criteria

We wanted to analyze the existing published results of papers at ICST'13, so we *included* papers which consisted of empirical studies of testing methods that published their data. We *excluded* other types of papers such as practical experience reports and papers with formal proofs.

D. Classification

We classified papers at several levels: their track, their focus, whether they published raw data, and whether the results were statistically significant. We considered a result statistically significant if MeansTest reported at least one significant result and MeansTest found at least as many significant results as the authors claimed; if MeansTest disagreed with the authors about the significance of their results, we classified that paper overall as not being statistically significant. In this way, we did not bias our classification against thorough experiments with many hypothesis tests and some statistically insignificant results.

E. Data extraction

There were four pieces of data we extracted from each paper: 1) the paper's hypotheses, 2) the hypothesis testing applied (if any), 3) the published results and claims of significance, and 4) the MeansTest assessment of the results. We extracted these through manual reading and copy/paste of tables. Whereas Dit et al. reproduced entire experiments and published the TraceLab components implementing them, we found that we could sufficiently answer our more modest research questions through meta-analysis of the published results.

Figure 2 shows the workflow we created to model individual comparisons of means in software testing experiments. Based on the paper's hypotheses, we manually established the nature of the samples (paired *vs.* two independent samples) as an input to the overall workflow. After inputting each sample, we executed the MeansTest workflow depicted in Fig. 1, abstracted here as the node labeled MeansTest. As an output, MeansTest provided information such as the p-value, the hypothesis test used, and the sample size required to get a significant result. We compared those results with the results the authors provided. We used this workflow to present the results below.

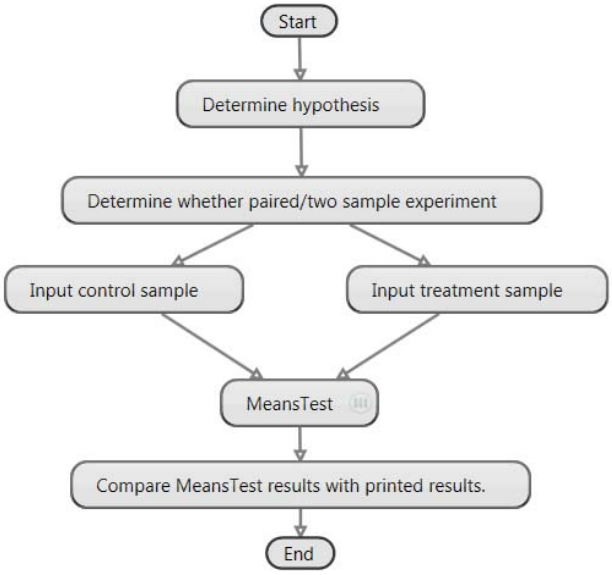

Fig. 2. Workflow for ICST 2013 meta-analysis.

VII. META-ANALYSIS RESULTS

Figure 3 shows the scope of the meta-analysis. There were **38** papers in the main testing and industry track. Of those, **24** papers (63%) featured some kind of empirical validation comparing a method with one or more baseline methods. Of those 24 papers, only eight papers (33%) reported the statistical significance of the authors' empirical validation. In response to **RQ1**, we note that **six papers (75%) used the Wilcoxon tests** with no consideration for alternative tests. Only one of these papers [20] printed enough raw data to enable a meta-analysis; the remaining 7 papers reported only summary statistics that we could not validate with MeansTest.

We examined the remaining 16 papers with no statistical validation for the presence of raw data. Of the 16 papers, 11 papers (69%) printed enough of their experiment data in the proceedings to suffice for a MeansTest meta-analysis. Together with the Canfora et al. paper, we analyzed **12** papers with data.

In response to **RQ2**, our meta-analysis found that **only six papers (50%)** had reproducible statistically significant results at the 95% confidence level. This proportion is consistent with the Kampenes et al. systematic review [17]. The remaining six papers invariably claimed "significant" results even though the experiment was too small to support the *statistical* significance of said results. These issues in the non-significant experiments could likely be remedied with a larger experiment. In the next sections, we use the MeansTest power analysis to recommend appropriate sample sizes.

A. Analyses by experiment

In this section, we briefly summarize the 12 experiments in question. We report the properties of the experiment as inferred by MeansTest, including normality, the appropriate hypothesis test, and MeansTest's p-value for the experiment. In cases where the authors' results were not statistically significant, we also state MeansTest's power analysis to suggest the appropriate course of action in order to get a significant result.

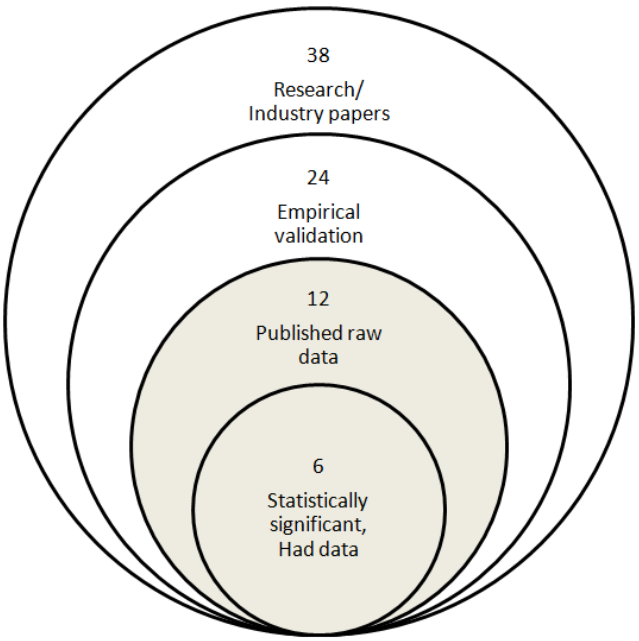

Fig. 3. Scope of ICST 2013 meta-analysis.

1) Multi-Objective Cross-Project Defect Prediction

Canfora et al. [20] introduce a regression model, which they call Multiple-Objective Logistic Regression, to predict defects across projects. They compare their model with a Within-Project Logistic model, a Single-Objective cross-project Logistic model, and a Clustering-Based Logistic model. They study 10 projects. For each project, they compute the cost of the model, its recall, and precision.

They concede that the Within-Project model **is better** than their cross-project model. **Using the Wilcoxon signed-rank test**, they report that the Multiple-Objective Logistic model **significantly diverges** from the Single-Objective Logistic model in terms of the cost (p=0.02), but not the precision (p=0.4). Finally, they report that the Multiple-Objective Logistic model **significantly diverges** from the Clustering-Based Logistic model in terms of the cost (p=0.009) but the precision is **borderline significant** (p=0.05). All of the models had identical recall.

This paper is of particular interest to this meta-analysis because it features raw data, existing statistical analysis of the results, and a statistically borderline p-value of 0.05. Better still, this inconclusive p-value was achieved because the authors used the least-powerful Wilcoxon test without justification. MeansTest was designed to address exactly this situation by automatically inferring whether the data is normal to lend more power to the analysis when appropriate.

Table I summarizes our meta-analysis of this paper. Most of our results were the same, but in the case of the borderline significant p-value, MeansTest concluded that the data was

sufficiently normally distributed to apply the t-test; this enabled the difference in precision to **become statistically significant**.

TABLE V. CANFORA ET AL. VS MEANSTEST

Logistic model	Cost p-value		Precision p-value	
	Author's	*MeansTest*	*Author's*	*MeansTest*
Within-Project	-	0.15	-	0.06
Single-Objective	0.02	0.02	0.4	1.0
Clustering-Based	0.009	0.01	**0.05**	**0.02**

In light of this small victory, one might pause to ponder whether the MeansTest p-values, *themselves*, are significantly different from the p-values of Canfora et al. and the broader research community. One could recursively apply MeansTest to the MeansTest p-values and the authors' p-values to make that determination. If the difference were not significant, the MeansTest power analysis would recommend the required sample size needed to get a significant difference. We leave this problem as future work.

2) Empirical Evaluation of the Statement Deletion Mutation Operator

Deng et al. [21] examined the effectiveness of the statement deletion mutation operator. They applied this mutation to 40 Java classes. For each class, the first author built a test set that killed every deletion mutant. They then applied the test set to muJava's mutants. They reported that the deletion operator used significantly **less** mutants than muJava to get **roughly the same** level of coverage as a test set generated from killing muJava mutants.

In applying MeansTest, we found that neither of the paired data sets were normally distributed. The statement deletion operator indeed produced **significantly less** mutants than muJava (Wilcoxon signed-rank, p~10^-8). However, MeansTest reported that the deletion operator had a **significantly worse** mutation score than a muJava test set with mutation score 1 (Wilcoxon signed-rank, p ~10^-7).

3) Symbolic Path-Oriented Test Data Generation for Floating-Point Programs

Bagnara et al. [22] introduced a performance optimization to the symbolic constraint solver for C code, FPSE. They ran their improved code against the stock code and measured the running time against 1-12 iterations of the C functions `dichotomic()` and `tcas_periodic_task_1Hz()`. They reported **improved execution times**, including solving some problems that caused the original code to time out.

MeansTest inferred that the performance data was approximately normal under `dichotomic()`, but not under `tcas_periodic_task_1Hz()`. The performance optimization was indeed **significantly faster** under `dichotomic()` (t-test, p=0.02) but **not significantly faster** under `tcas_periodic_task_1Hz()` (Wilcoxon signed-rank, p-value=0.12). According to MeansTest's power analysis, the authors would need to run at least **55 iterations** to get a statistically significant difference in performance.

4) Generating Effective Integration Test Cases from Unit Ones

Pezzè et al. [23] developed an Eclipse plugin, called Fusion, for automatically generating integration test cases from the semantics of unit test cases. They compared the number of faults and false positives found by their method with two other tools: Randoop and Palus. They performed this comparison across four programs. They reported that Fusion found **different faults** than Randoop and Palus, but had a **comparable number** of false positives.

We applied MeansTest 4 times total to compare Fusion with the other two methods. MeansTest inferred that the differences between the methods were normally distributed. It is difficult to formulate a hypothesis for assessing the statistical significance of finding "different" faults, but Fusion **did not find** significantly different number of real faults than either Randoop or Palus (t-test, p=0.09 and 0.13, respectively). Fusion indeed found about the same false positives as Randoop and Palus (t-test, p=0.26 and 0.12, respectively). According to MeansTest's power analysis, the authors would need to test at least **12 programs** to notice a difference in real faults, and **26 programs** to notice a difference in false positives.

5) Improving Test Generation under Rich Contracts by Tight Bounds and Incremental SAT Solving

Abad et al. [24] developed a new test generator, called FAJITA. They compared the branch coverage and performance of FAJITA with Pex, Kiasan, Randoop, AutoTest, and EvoSuite. They ran these tools on 25 methods across 8 classes. They reported that FAJITA **had the best branch coverage** of all tools.

We applied MeansTest 4 times to compare FAJITA's branch coverage to that of each of the other tools. We configured MeansTest to state the authors' hypothesis as one-sided. MeansTest inferred that the differences between the tools were not normally distributed. FAJITA did have **significantly greater** branch coverage than Pex, Kiasan, Randoop, AutoTest, and EvoSuite (Wilcoxon signed-rank, p=0.0008, 0.03, 0.0002, 0.0004, 0.01, respectively).

6) Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems

Kapfhammer et al. [25] developed an input generator, called AVM, to test the constraints on database schemas. They compared the constraint coverage of AVM to DBMonster. They reported that AVM had **better constraint coverage** than DBMonster.

MeansTest inferred that the difference between the data was not normally distributed. MeansTest concluded that the constraint coverage of AVM was **significantly better** than DBMonster (Wilcoxon, p~10^-5).

7) MFL: Method-Level Fault Localization with Causal Inference

Shu et al. [26] applied spectrum-based fault localization at the method level. They compared their technique, MFL, to existing SBFL tools Tarantula, Ochiai, PFIC, and one based on the F-measure. They ran these tools across 4 programs each seeded with about 7 faults and calculated the minimum cost of

a developer searching through methods according to the suspiciousness ranks to find a bug. The authors reported that MFL **was cheaper** to use than the other measures in 3 out of 4 programs.

MeansTest inferred that the difference between the minimum costs of the methods was normally distributed. MeansTest concluded that the minimum cost of MFL was **significantly cheaper** than Tarantula and PFIC (t-test, p=0.033 and 0.34, respectively), but **not significantly** cheaper than Ochiai and F-measure (t-test, p=0.07 and 0.06, respectively). According to MeansTest's power analysis, the authors would need to test at least **11 programs** to notice a difference in minimum cost in all four tools.

8) Scaling Model Checking for Test Generation using Dynamic Inference

Yeolekar et al. [27] developed a test generation tool, called AutoGen, for satisfying structural coverage criteria. They compared the branch coverage of test cases generated with their tool against random testing and another test generator called SatAbs. They applied these tools to 10 functions. They reported that AutoGen had **better coverage** than SatAbs and random testing.

AutoGen had **significantly higher** coverage than random (t-test, p~10^-5). The analysis of the difference between SatAbs and AutoGen is tricky because although SatAbs had higher coverage in some instances, it timed out on most functions. If we treat the timeouts as 0% coverage, we see that AutoGen had **significantly higher** coverage than SatAbs, but the difference was not normally distributed (Wilcoxon signed-rank test, p= 0.04).

9) Transformation Rules for Platform Independent Testing: An Empirical Study

Eriksson et al. [28] introduced UML transformations to identify implicit logical predicates ahead of time, before code is generated from the models. Their goal was to reduce the number of requirements needed to satisfy logic coverage criteria such as all-pairs and MCDC. They examine the UML of 6 programs and apply their transformations to the programs. They then compute the number of new requirements generated going from UML to code and show that their method requires **less** new rules.

MeansTest inferred that the data was not normally distributed. The implicit-to-explicit transformations did indeed generate **significantly less** rules going from the UML to code; this result applied to both all-pairs and MCDC coverage requirements (Wilcoxon, p=0.008).

10) An Efficient Algorithm for Constraint Handling in Combinatorial Test Generation

Yu et al. [29] introduced their combinatorial test generation tool, called ACTS. They compared ACTS to other combinatorial test generators: CASA, Ttuples, and PICT. They compared the tools' performance in terms of the amount of time spent building the test set. They evaluated their tools across 16 programs and concluded that "ACTS can perform significantly **better** for systems with more complex constraints."

MeansTest inferred that the performance data was not normally distributed. ACTS was **significantly faster** than CASA and TTuples, but **not** PICT (Wilcoxon, p=0.0001, 0.004, and 0.37, respectively). MeansTest estimates that the authors would need **35 programs** to show a statistically significant difference in the runtime performance between ACTS and PICT.

11) Oracle-Based Regression Test Selection

Yu et al. [30] examined the problem of regression test selection as part of change impact analysis at ABB. They discussed two broad methods of creating test oracles: using outputs and tracking the internal state. They introduced an algorithm for inferring the test cases needed to test a change, based on so-called "internal oracles" that study the effect of changes on the internals of a system. They compared the faults found by test sets selected by internal oracles with those generated by "output oracles" (oracles that only check the output) on 9 programs. They found that internal oracles discover **significantly more** faults than output oracles.

MeansTest inferred that the fault distribution data was normally distributed. The internal oracle tests found **significantly more** faults than the output oracle tests (t-test, p=0.002).

12) Test Case Prioritization Using Requirements-Based Clustering

Arafeen and Do [31] examined the issue of test case prioritization: which test cases are most likely to uncover faults? They introduced a new test case clustering technique that orders test cases based on the priority of their requirements. They introduce several within-cluster ordering heuristics as well. They compare the effectiveness of prioritizing with clustering against standard McCabe-style prioritization metrics on four programs: three versions of iTrust and Capstone. They find that clustering **outperforms** McCabe on iTrust, but **not** on Capstone.

MeansTest inferred that the relative effectiveness percentages were normally distributed. The clustering technique **significantly outperformed** McCabe on the iTrust code (t-test, p~10^-14) but **not** on Capstone (t-test, p=0.18). The Capstone program was too small to perform a meaningful analysis.

VIII. DISCUSSION AND FUTURE WORK

An automated selection process for statistical analysis can help researchers draw conclusions about the statistical significance of their results in the absence of an expert statistician. Our validation showed that this process significantly outperformed blind adherence to the Wilcoxon nonparametric tests. In light of newer nonparametric hypothesis tests such as Brunner-Munzel, the adherence to the Wilcoxon tests that prevailed in ICST'13 may be outdated. Indeed, we found a specific instance at ICST'13 where our process found a significant result that was originally reported as being of questionable significance. While very promising, our results indicate that our process is still not perfect; it is not a substitute for an expert statistician. It is ultimately up to an expert to decide which test is most appropriate in a given situation.

Cross-referencing our meta-analysis with the systematic review by Kampenes et al. [17], we see that ICST'13 had almost identical statistical significance as journal papers. About 50% of results were statistically significant in both studies. Unfortunately, authors at ICST'13 under-reported the statistical significance of their work compared to the Kampenes et al. journal papers, with only 33% of empirical validation papers at ICST'13 reporting their statistics. We hope that our workflow will make it more convenient for authors in the future to report statistical significance.

In our meta-analysis of ICST'13, we stated the minimum sample sizes required to achieve statistically significant results. Our methodology, well-known to statisticians as power analysis, is a welcome addition to the automated statistics mode of thought. Many problem domains call for even more sophisticated analysis, such as blocked designs and analysis of variance, for which we have yet to provide a solution. There are several models of analysis of variance, each with their own assumptions, so this type of analysis would benefit from an automated selection process similar to that of MeansTest and remains as future work.

ACKNOWLEDGMENT

This work was funded in part by the National Science Foundation under grant CCF-0811140 (research) and Major Research Instrumentation grant CNS-0959924 provided by ARRA funding.

REFERENCES

[1] D. D. Nardo, N. Alshahwan, L. C. Briand, and Y. Labiche, "Coverage-Based Test Case Prioritisation: An Industrial Case Study," in *ICST*, 2013, pp. 302–311.

[2] *2013 IEEE Sixth International Conference on Software Testing, Verification and Validation, Luxembourg, Luxembourg, March 18-22, 2013*. IEEE, 2013.

[3] J. Cohen, *Statistical power analysis for the behavioral sciences*, 2nd ed. Hillsdale, N.J: L. Erlbaum Associates, 1988.

[4] Student, "The Probable Error of a Mean," *Biometrika*, vol. 6, no. 1, p. 1, Mar. 1908.

[5] F. Wilcoxon, "Individual Comparisons by Ranking Methods," *Biom. Bull.*, vol. 1, no. 6, p. 80, Dec. 1945.

[6] E. Brunner and U. Munzel, "The Nonparametric Behrens-Fisher Problem: Asymptotic Theory and a Small-Sample Approximation," *Biom. J.*, vol. 42, no. 1, pp. 17–25, Jan. 2000.

[7] K. Pearson, "' Das Fehlergesetz und Seine Verallgemeinerungen Durch Fechner und Pearson.' A Rejoinder," *Biometrika*, vol. 4, no. 1/2, pp. 169–212, 1905.

[8] S. S. Shapiro and M. B. Wilk, "An analysis of variance test for normality (complete samples)," *Biometrika*, vol. 52, no. 3–4, pp. 591–611, Dec. 1965.

[9] M. B. Brown and A. B. Forsythe, "Robust Tests for the Equality of Variances," *J. Am. Stat. Assoc.*, vol. 69, no. 346, pp. 364–367, Jun. 1974.

[10] R Development Core Team, "R: A Language and Environment for Statistical Computing," 2008. [Online]. Available: http://www.R-project.org.

[11] M. Hays, J. H. Hayes, A. Stromberg, and A. Bathke, "Traceability Challenge 2013: Statistical Analysis for Traceability Experiments Software Verification and Validation Research Laboratory (SVVRL) of the University of Kentucky"," in *Proceedings of Traceabillity of Emerging Forms of Software Engineering 2013*.

[12] "TraceLab." [Online]. Available: http://coest.org/index.php/about-coest8/current-projects/tracelab. [Accessed: 23-Jan-2012].

[13] N. M. Razali and Y. B. Wah, "Power comparisons of Shapiro-Wilk, Kolmogorov-Smirnov, Lilliefors and Anderson-Darling tests," *J. Stat. Model. Anal. Vol*, vol. 2, no. 1, pp. 21–33, 2011.

[14] B. Dit, E. Moritz, M. Linares-Vasquez, and D. Poshyvank, "Supporting and Accelerating Reproducible Research in Software Maintenance using TraceLab Component Library," in *Proceedings of 29th IEEE International Conference on Software Maintenance (ICSM'13)*, Eindhoven, the Nethelands, 2013.

[15] K. Petersen, R. Feldt, S. Mujtaba, and M. Mattsson, "Systematic mapping studies in software engineering," in *Proceedings of the 12th international conference on Evaluation and Assessment in Software Engineering*, Swinton, UK, UK, 2008, pp. 68–77.

[16] V. Basili, R. Selby, and D. Hutchens, "Experimentation in Software Engineering," *IEEE Trans. Softw. Eng.*, vol. SE-12, no. 7, pp. 733–743, Jul. 1986.

[17] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. K. Sjøberg, "A systematic review of effect size in software engineering experiments," *Inf. Softw. Technol.*, vol. 49, no. 11–12, pp. 1073 – 1086, 2007.

[18] R. R. Wilcox, *Applying contemporary statistical techniques*. Amsterdam ; Boston: Academic Press, 2003.

[19] R. Baeza-Yates and B. Ribeiro-Neto, *Modern information retrieval*. New York; Harlow, England: ACM Press ; Addison-Wesley, c1999., 1999.

[20] G. Canfora, F. Mercaldo, C. A. Visaggio, M. D'Angelo, A. Furno, and C. Manganelli, "A Case Study of Automating User Experience-Oriented Performance Testing on Smartphones," in *ICST*, 2013, pp. 66–69.

[21] L. Deng, J. Offutt, and N. Li, "Empirical Evaluation of the Statement Deletion Mutation Operator," in *ICST*, 2013, pp. 84–93.

[22] R. Bagnara, M. Carlier, R. Gori, and A. Gotlieb, "Symbolic Path-Oriented Test Data Generation for Floating-Point Programs," in *ICST*, 2013, pp. 1–10.

[23] M. Pezzè, K. Rubinov, and J. Wuttke, "Generating Effective Integration Test Cases from Unit Ones," in *ICST*, 2013, pp. 11–20.

[24] P. Abad, N. Aguirre, V. S. Bengolea, D. Ciolek, M. F. Frias, J. P. Galeotti, T. Maibaum, M. M. Moscato, N. Rosner, and I. Vissani, "Improving Test Generation under Rich Contracts by Tight Bounds and Incremental SAT Solving," in *ICST*, 2013, pp. 21–30.

[25] G. M. Kapfhammer, P. McMinn, and C. J. Wright, "Search-Based Testing of Relational Schema Integrity Constraints Across Multiple Database Management Systems," in *ICST*, 2013, pp. 31–40.

[26] G. Shu, B. Sun, A. Podgurski, and F. Cao, "MFL: Method-Level Fault Localization with Causal Inference," in *ICST*, 2013, pp. 124–133.

[27] A. Yeolekar, D. Unadkat, V. Agarwal, S. Kumar, and R. Venkatesh, "Scaling Model Checking for Test Generation Using Dynamic Inference," in *ICST*, 2013, pp. 184–191.

[28] A. Eriksson, B. Lindström, and J. Offutt, "Transformation Rules for Platform Independent Testing: An Empirical Study," in *ICST*, 2013, pp. 202–211.

[29] L. Yu, Y. Lei, M. N. Borazjany, R. Kacker, and D. R. Kuhn, "An Efficient Algorithm for Constraint Handling in Combinatorial Test Generation," in *ICST*, 2013, pp. 242–251.

[30] T. Yu, X. Qu, M. Acharya, and G. Rothermel, "Oracle-based Regression Test Selection," in *ICST*, 2013, pp. 292–301.

[31] M. J. Arafeen and H. Do, "Test Case Prioritization Using Requirements-Based Clustering," in *ICST*, 2013, pp. 312–321.