2013 ACM / IEEE International Symposium on Empirical Software Engineering and Measurement

A Replicated Experiment on the Effectiveness of Test-first Development

Davide Fucci
Department of Information Processing Science
University of Oulu
Oulu, Finland
Email: davide.fucci@oulu.fi

Burak Turhan
Department of Information Processing Science
University of Oulu
Oulu, Finland
Email: burak.turhan@oulu.fi

***Abstract*—Background: Test-first development (TF) is regarded as a development practice that can lead to better quality of software products, as well as improved developer productivity. By implementing unit tests before the corresponding production code, the tests themselves are the main driver to such improvements. The role of tests on the effectiveness of TF has been studied in a controlled experiment by Erdogmus et al. (i.e. original study).**
Aim: Our goal is to examine the impact of test-first (TF) development on product quality and developer productivity, specifically the role that tests play in it.
Method: We replicated the original study's controlled experiment by comparing an experimental group applying TF to a control group applying a test-last approach. We then carried out a correlation study in order to understand whether the number of tests is a good predictor for external quality and/or productivity.
Results: Mann-Whitney tests did not show any significant difference between the two groups in terms of number of tests written (W=114.5, p=0.38), developers' productivity (W=90, p=0.82) and external quality (W=81.55, p=0.53). In addition, while a significant correlation exists between the number of tests and productivity (Spearman's ρ = 0.57, p$<$0.001), none was found in the case of external quality (Spearman's ρ = 0.17, p=0.18).
Conclusions: We conclude that TF neither improves nor deteriorates the external quality or the productivity when compared to the test-last approach, leaving room for other variables to impact the effects of TF. This replication has partially confirmed the findings of the original study.

I. Introduction

Test-first development (TF), also referred to as Test-driven development (TDD)[1], is a software development practice that has attracted interest among the industry professionals as well as in the academic curricula in the last decade. TF enforces short development cycles, in which tests are written before the actual functionality, and then the minimal amount of production code necessary to pass the test is written. Finally, the code is refactored if necessary [1]. This shift in the approach to software development is claimed to improve code quality in terms of bugs detection rate and internal design, and improve programmers' productivity [2]. The claims made by TF development proponents have been the subject of several studies in recent years. Some of those studies (see Section II) used controlled experiments as research method and compared groups of TF developers to non-TF developers in order to build the evidence necessary to verify or refute the claims. The results across studies have been inconclusive so far [3], [4]. In our study we focus on the analysis of tests as a factor that might lead to the claimed effects. The role of tests has been often neglected during the previous investigations of TF development except for the work of Erdogmus et al. [5], in which the number of tests is used to predict external quality and productivity. Indeed, the number of tests and the test-to-production code ratio are the main differentiating points between TF and more classical development practices [5]. In this paper we present a *close replication* [6], also referred to as *exact replication* [7], of Erdogmus et al.'s experiment, henceforth *the original study*. Through this replication we checked the validity of the model presented in the original study, and expanded it by analyzing the impact of the necessary changes we had to make in the experimental settings.

[1]Please note that for the scope of this study we consider test-first as a synonymous of test-driven development

The paper is organized as follows: Section II presents a summary of the previous studies regarding TF development. Section III describes the original study. Section IV presents our replication with goals, context, data analysis and hypotheses testing while the threats to validity are discussed in Section V. In section VI our results are discussed and compared to the original study. Finally, Section VII presents the conclusion and addresses future work.

II. Related Work and Background

In this section, we first report the results of two recent secondary studies regarding the effects of TF on productivity and external quality, then briefly analyze eleven controlled experiments reported in the included primary studies, and finally present the findings from qualitative studies.

The systematic review by Turhan et al. [8] gathered evidence among studies in industry and academia between 2000 and 2008. The overall picture is that the effects of TF are slender. In the 32 primary studies — published in 22 peer-reviewed papers — reporting a comparison between TF and test-last development (TL), external quality seems to be improved by TF while inconclusive results are obtained for productivity. A similar result is reported in a meta-analysis by Rafique and Misic [9]. TF development modestly increases quality while the effect on productivity are not clear. The 37 primary studies — published until February 2011 in 25 peer reviewed papers — were analyzed according to different factors, such as the methodology followed by the control

978-0-7695-5056-5/13 $26.00 © 2013 IEEE
DOI 10.1109/ESEM.2013.15

group (waterfall vs. iterative test last), the subject's experience, the use of other XP practices like pair-programming, and the context (industrial vs. academic). Software quality was improved under the industrial settings, although no significant correlation was shown between quality and experience of the subjects. Interestingly, in the industrial setting a drop in productivity was observed. One reason could be that experienced subjects strictly adhere to TF approach and spend more time on refactoring and unit-testing. Finally, a correlation between the use of other agile methodologies and improved productivity was not observed [9].

Throughout several studies using controlled experiments, the findings about the effects of TF on productivity and external quality are controversial. Some studies [10]–[12] show that the effect on productivity are positive, while others [13]–[15] show the contrary. A third group of studies did not show any significant difference between the two techniques [16]–[18]. Interestingly, TF is associated with better productivity when accompanied with pair-programming [11], [12]. The effects of TF on external quality are reported to be positive in industrial studies [19], [20], as well as in academic settings [10], [12]. Nevertheless, more rigorous studies [16], [18] report inconclusive results. The perception of the effects of TF development on quality and productivity has also been gauged through survey and questionnaire studies.

A survey among 240 Canadian computer science students [21], ranging from undergraduates to Ph.D.'s, about extreme programming methodologies showed that 70% agreed that TF development improved external quality and productivity. 55% of the respondents declared that they used, or are willing to use, TF in their next projects. Remarkably, the favor for TF was expressed mostly by more experienced subjects. A similar result is reported in a study by Janzen and Saiedian [22], which surveyed 130 novices and 30 experienced programmers. 70% of the latter believed that TF improved external quality and productivity, and 90% believed that it produces better designed and testable code as well. Nonetheless, such opinion was shared by only 30% of less experienced subjects. Finally, the results of a questionnaire administered during a study by Flohr and Schneider [23] showed that TF neither improves or worsen quality (85% of the respondents), nor productivity (70% of the respondents) compared to a traditional development approach. On the other hand, 73% of the respondents supported the statements that *"Test-first helps not to write more code than necessary"* and 63% agreed that *"Test-first helps to write code with less flaws"*.
This study is the first attempt, to the best of our knowledge, to replicate a controlled experiment about the effects of TF on external quality and productivity. The guidelines proposed by Carver [24] are used to report our replication.

III. Original Study

Due to space limitations, we briefly describe the original experiment in this section. The original study included two stages. In the first stage, the test-first (TF) approach was compared to the test-last (TL) through a controlled experiment. In particular the authors hypothesis was that TF development has positive effects on the number of unit-test written by the developers, the external code quality and the developers' productivity. In the second stage, the authors studied the correlation between the number of tests — a central feature of the TF approach — and the external code quality and productivity. The rationale was that if the TF approach encourages developers to write more tests and that there is a positive correlation between the number of tests and quality and productivity, then TF would improve the overall quality and productivity. The research questions of the original study and the related formalized hypotheses, presented in Table I, along with the results, are as follows [5]:

1) Do test-first programmers write more tests than test-last programmers ? (Hypothesis 1T in Table I).
2) Do test-first programmers produce higher quality than test-last programmers? (Hypothesis 1Q in Table I).
3) Are test-first programmers more productive than test-last programmers? (Hypothesis 1P in Table I).
4) Does a higher number of tests imply higher quality? (Hypothesis 2Q in Table I).
5) Does a higher number of tests imply higher productivity? (Hypothesis 2P in Table I).

The context of the original study is summarized in Table II. The hypothesis were tested using Mann-Withney U-test with $\alpha = 0.1$, the effect size was calculated using Cohen's d but has not been reported. The correlations were calculated using Spearman ranks correlation.

IV. Replication

A. Motivation

This replication was carried out following the framework devised by Juristo and Vegas [6]. In order to verify or refute the model presented in the original study and check its robustness, we introduced new conditions and applied it to a new dataset. The definition of the experiment according to Wohlin et al. template [25] is:
The *object* of the study is unit-tests.
The *purpose* is to evaluate the impact of TF development on unit-tests.
The *quality focus* is external code quality and programmers' productivity.
The *perspective* is from the researchers' point of view.
The *context* is a Java lab exercise session with graduate and undergraduate students.

B. Research questions, Metrics and Hypotheses

Our research questions include the same as in the original study (see Section III). Nevertheless, we formulated an additional research question related to the effects of having mixed subject units (i.e. pair and individual programmers).
RQ6: Does the subject's unit interact with the outcome?
Along with the three variables used in the original experiment (i.e. $TESTS$, $QLTY$, $PROD$) we introduced a new variable, $SUBJ$. The metrics we used to measure each variable are described as follows:
$TESTS$: The number of tests developed as a single JUnit test case by the subject. The range of this ratio variable is $[0, \infty)$.
$QLTY$: Defect-based external quality achieved by the subject, defined as the average quality of the delivered user stories. It is the number of passed acceptance tests over the total number of tests. This variable range is [0.5, 1].

TABLE I: Original study hypotheses and outcomes (from [5])

Stage	Name	H_0	H_1	Outcome
1	1T	$\#TESTS(TF) = \#TESTS(TL)$	$\#TESTS(TF) > \#TESTS(TL)$	Reject H_0
1	1Q	$QUALITY(TF) = QUALITY(TL)$	$QUALITY(TF) > QUALITY(TL)$	Failed to reject H_0
1	1P	$PROD(TF) = PROD(TL)$	$PROD(TF) > PROD(TL)$	Failed to reject H_0
2	2Q	$QUALITY = \beta_0 + \beta_1 \times \#TESTS, \beta_1 = 0$	$QUALITY = \beta_0 + \beta_1 \times \#TESTS, \beta_1 \neq 0$	Failed to reject H_0
2	2P	$PROD = \beta_0 + \beta_1 \times \#TESTS, \beta_1 = 0$	$PROD = \beta_0 + \beta_1 \times \#TESTS, \beta_1 \neq 0$	Reject H_0

TABLE II: Original study and replication settings

	Original	Replication
Subject Type	35 Undergraduate (11 dropped)	33 graduate, 25 undergraduate
Subject Unit	Individuals	11 pairs, 36 individuals
Subject Environment	Java course	Software quality and testing course (15 hours hands-on lab course)
Subject Stratification	Yes, according to skill (low, medium, high)	None
Programming Environment	Java, Eclipse, JUnit	Java, Eclipse, JUnit
Experiment task	Robert Martin's Bowling Scorekeeper	Robert Martin Bowling Scorekeeper
Task type	Fine grained, incremental difficulty	Fine grained, incremental difficulty
Time to complete the task	Several lab sessions, remote work	Single lab session (3 hours)
Experiment design	One factor, two treatments	One factor, two treatments
Experiment groups	test-first vs. test-last	test-first vs. test-last
Control / Treatment size	13 / 11	20 (4 pairs, 16 individuals) / 27 (7 pairs, 20 individuals)
Variables	$TESTS, PROD, QLTY$	$TESTS, PROD, QLTY, SUBJ$

$PROD$: The number of the delivered user stories by the subject divided by the total number of user stories. This variable range is [0, 1].
$SUBJ$: A categorical variable indicating whether the subject was an individual (I) or a pair (P).
Our research hypotheses include the ones formulated in the original study (see Table I). In addition, two hypotheses — to check for the interaction of the subject unit — are formulated as follows:
$2S_{2P}$ - The subject unit does not have an interaction with the correlation expressed in $2P$.

H_{0-2Sa} - $\beta_1(I) = \beta_1(P)$
H_{1-2Sa} - $\beta_1(I) \neq \beta_1(P)$
H_{0-2Sb} - $\beta_0(I) = \beta_0(P)$
H_{1-2Sb} - $\beta_0(I) \neq \beta_0(P)$

$2S_{2Q}$ - The subject unit does not have an interaction with the correlation expressed in $2Q$.

H_{0-2Sa} - $\beta_1(I) = \beta_1(P)$
H_{1-2Sa} - $\beta_1(I) \neq \beta_1(P)$
H_{0-2Sb} - $\beta_0(I) = \beta_0(P)$
H_{1-2Sb} - $\beta_0(I) \neq \beta_0(P)$

C. Design

It is important to note that $1T$, $1Q$ and $1P$ are experimental hypotheses, while hypotheses $2Q$, $2P$, $2S_{2Q}$ and $2S_{2P}$ are non-experimental, but related to the correlation study, in which β represents the regression coefficient [25].

The experiment is originally designed and carried out with one factor — the software development methodology — and two alternative treatments, TF and TL development. Due to changes in the experiment context a second factor, namely the subject unit, was unintentionally introduced due to the restrictions imposed by the context (i.e. limited amount of workstations available) and therefore it was not possible to have full control on it. We decided to follow the original experimental design, and used the subject unit in further analysis of the model. No systematic blocking was applied. Although the original study suggested to use the subject's skill level as a blocking variable, our data (e.g. the pre-questionnaire results, quality of artifacts developed during the previous lab sessions before the experimental task) was collected anonymously and therefore gave us only an overall view of the subjects' skill level. We are aware that factors such as skill and experience might have and impact on the outcome, and we note our concerns in Section V. Even though it is preferable to have balanced experimental groups [25], due to the experiment setting and apparatus, our design was unbalanced.

D. Subjects and Sampling

The subjects — chosen on convenience — are senior undergraduate and fist year graduate students from the department of Information Processing Science at the University of Oulu, Finland. The subjects took a graduate-level course about software quality and testing during the Fall term of 2012. The course was accompanied by six lab sessions of three hours each. During the lab sessions, the subjects learned how to use the Eclipse IDE to develop unit tests with Java and JUnit, the refactoring tools included in the IDE and TF development. The subjects were required to use TF to solve several code katas [26] of increasing difficulty throughout the sessions. The experiment sample included the subjects present during the last lab session, when the experimental task was tackled, hence the experiment was run offline [25]. The 70 students enrolled in the course were randomly divided into TF and TL groups before the experiment started. Nevertheless, the subjects who actually attended the experiment session were only 58 (33 graduate and 25 undergraduate) divided into 11 pairs and 36 individuals. Of those 47 units, 20 had been assigned to the TL group and 27 to the TF group. The TL group was composed of 4 pairs and 16 individuals, while the TF group was composed of 7 pairs and 20 individuals, leading to an unbalanced design. The subjects agreed to participate to the experiment by signing and informed consent form. To assess the subjects' skills and experience

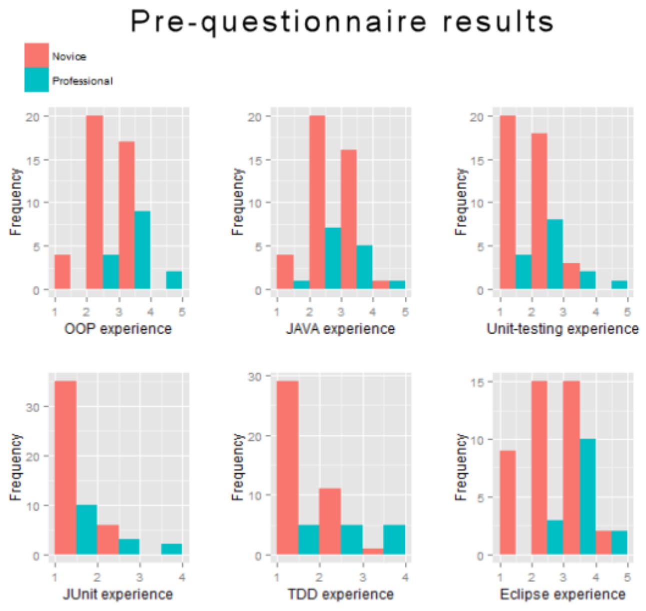

Fig. 1: Pre-questionnaire results divided according to subjects' experience

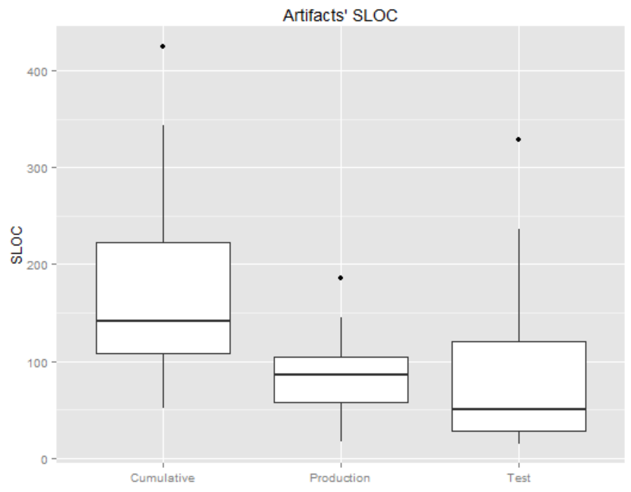

Fig. 2: Boxplots for cumulative, production code and test code SLOC of the software artifacts.

they were asked to fill in an anonymous pre-questionnaire. In the questionnaire we asked the subjects to indicate if they had professional experience in software development. Two subjects did not fill in the questionnaire, therefore we retained 56 answers: 15 professionals (i.e. industrial experience), 41 novices. The overall results are presented in Figure 1. The professionals' experience varied between 3 months and 10 years with an average of 2.6 years. Although the subjects were informed about their participation to the experiment, the research questions were not disclosed to them in advance.

E. *Objects and Artifacts*

The subjects tackled the experiment task during the last lab session, when only 19 workstations were available. We divided the final session into three consecutive three-hour sessions in order to allow all subjects to undertake the task. We made sure that there was no interaction between the subjects from different sessions; therefore we are confident that such setting did not pose any threat to internal validity. The experimental task was a modified version of Martin's Bowling Scorekeeper code kata utilized by George and Williams [15] in their experiment as well as in the original study. The task was divided into 13 fine-grained user stories of increasing difficulty, each containing an example of the input and the associated expected output. We provided the subjects a stub project to get them started on the development of the task. The stub project contained the methods' signatures that represented an API later used to run acceptance tests (30 LOC) and a test class, containing a single *dummy* unit-test as reference (15 LOC). We developed an acceptance test suite, which included 56 black-box tests, to collect external quality metrics. Figure 2 plots the size of the software artifacts produced by the subjects.

Finally, the pre-questionnaire — used to have a qualitative idea of the subjects' skill and experience — and a post-questionnaire to collect the subjects' opinions, were also part of the experimental package. The tasks used during the course, the questionnaires and the acceptance test suite are available for further replication[2].

Table II summarizes the context of the replication.

F. *Level of Interaction*

The authors of the original study did not take part in the replication process actively; therefore this replication is to be considered external [7]. The original investigators provided us with a laboratory package containing the spreadsheet used to calculate and report their results, the description of the experimental task and the acceptance test suite used to calculate the metrics. The laboratory package lacked the information about the difficulty of user stories in the experimental task, the artifacts delivered by the subjects and the CVS log files used to calculate the programming effort of each subjects. We used the spreadsheet to check the correctness of the formula we used to calculate the metrics, the acceptance test suite provided was used to calculate the metrics, even though some changes were necessary (see Section IV-G2).

G. *Changes to the Original Study*

When comparing our replication to the original study we identified two type of changes: in the experimental context and in the metrics.

1) Context: We identified two changes between the context of the original experiment and this replication:

(a) The time required to implement the exercise was fixed (3 hours).

[2]http://cc.oulu.fi/~dfucci/lab_package2012.zip

(b) Some subjects tackled the task using pair programming.

Change a) was known before the study took place due to the course organization, since the lab rooms could be booked only for a short time. Nevertheless, the new setting allowed us a higher level of control. This change has an impact on the metrics used to measure tests, productivity and quality. We expected that the short time span to complete the task would have an impact on the subjects. In particular, in the case of the TF group, the subjects might deviate from the process in order to complete the assignment. As a consequence of the shorter time required to complete the task, we simplified the original task by removing the requirements dealing with handling the user input and formatting the output. We made this change in order to balance the effects of time pressure on the subjects. The impact of this change is discussed in Section V.

Change in b) was necessary due to the limited number of workstations available in the lab. The subjects were familiar with pair programming since they had practiced it during the previous lab sessions. We checked the impact of pair programming by testing hypotheses $2S_{2P}$ and $2S_{2Q}$.

2) Metrics: The limited time necessary to complete the task had an impact on the metrics. Therefore, the formula used to calculate the value for $TESTS$, $QLTY$ and $PROD$ had to be changed. In particular, the original study used:

$$TESTS = \frac{\text{number of tests}}{\text{total programming effort}} \tag{1}$$

While in our case $TESTS$ corresponds to the number of tests, since the total programming effort — the time used to complete the task — was fixed.

The metric for $PROD$ in the original study was calculated as:

$$PROD = \frac{\text{number of delivered user stories}}{\text{total programming effort}} \tag{2}$$

We needed to adjust this measure and therefore our measure for $PROD$ is the percentage of delivered user stories.

The metric for $QLTY$ in the original study was calculated as:

$$QLTY = \frac{\sum_{i=1}^{\#u.s.} QLTY_i \times difficulty_i}{\#u.s.} \tag{3}$$

Where $\#u.s.$ is the total number of delivered user stories, and the quality of the *i*-th user story is defined as:

$$QLTY_i = \frac{\text{passing assert statements from the associated test}}{\text{total assert statements in the associated test}} \tag{4}$$

Formula 3 took into account the difficulty of each user story. Since we were not able to get information about how such difficulty parameter was calculated, we decided to be conservative and to exclude it, while leaving unchanged the rest of the formula. It is important to note that for the original metric a user story was as well considered delivered if it had passed at least 50% of the associated tests in the acceptance test suite. Some of the acceptance tests provided by the original investigators were meaningless for our simplified version of the task, in particular the acceptance test cases dealing with the validation of the user input and presentation of the output. Therefore, in our replication, the acceptance test suite included 56 tests out of the 105 present in the original one. Although

TABLE III: One datapoint was removed since none of the user stories was delivered, yielding a division by zero when calculating $QLTY$. The new values, without the datapoint, are given in parentheses.

Metric	Mean	StdDev	Max	Median	Min
		Cumulative — n=47 (n=46)			
TESTS	6.2 (6.3)	4.83 (4.83)	20 (20)	5 (5)	1 (1)
PROD	0.34 (0.33)	0.24 (0.24)	0.92 (0.92)	0.23 (0.23)	0.00 (0.08)
QLTY	0.89 (0.89)	0.05 (0.05)	1.00 (1.00)	0.89 (0.89)	0.78 (0.78)
		TF group — n=27			
TESTS	7.15	5.17	20	7.15	1
PROD	0.35	0.26	0.92	0.23	0.08
QLTY	0.90	0.05	1.00	0.89	0.78
		TL group — n=20 (n=19)			
TESTS	4.9 (5.11)	4.13 (4.14)	14 (14)	4 (4)	1.00 (1.00)
PROD	0.34 (0.36)	0.23 (0.22)	0.85 (0.85)	0.23 (0.23)	0.00 (0.08)
QLTY	0.84 (0.89)	0.20 (0.05)	0.85 (1.00)	0.89 (0.89)	0.00 (0.78)

we expect this change to have no impact, since the new metrics remained consistent in this study, the different metrics will not allow a direct comparison between the original study and ours.

H. Analysis

In this section we present a summary of the statistical analysis process that led to our results.

1) Descriptive Statistics: Descriptive statistics of the metrics are summarized in Table III. The first observation is that the two groups performed similarly in terms of quality and productivity, while there is a slight difference in terms of the number of unit-tests implemented.

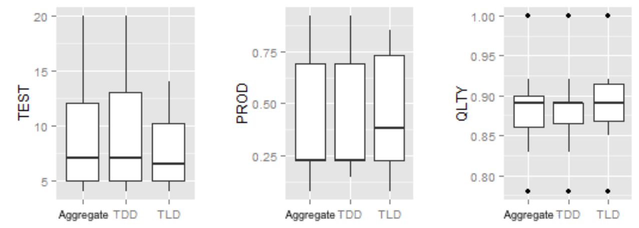

Fig. 3: Boxplots for number of tests, productivity and quality.

2) Dataset Reduction: Before testing our hypotheses, we assessed the quality of the data, and checked whether the subjects reasonably participated in the experiment. We eliminated the datapoint for which $PROD = 0$ and $QLTY$ could not be calculated (see Table III). One clue about the active participation of the subjects during the task is the number of lines of code written. Another factor indicating whether they were engaged in the task is the number of tests they wrote, since the focus of the course was unit-testing. We extracted from the dataset all the datapoints for which the production LOC and $TESTS$ were below their respective 1st quantiles (57.5 for LOC, 2.25 for $TESTS$); obtaining 18 datapoints candidate to be removed. After manual inspection of the artifacts associated with those datapoints, we decided to definitively remove them since they were constituted by stubs methods (e.g. methods returning a constant value) or empty methods (both in production or test code).

TABLE IV: Summary for test of hypotheses $1T, 1P$ and $1Q$.

Hypothesis (Variable)	W-statistic	p-value	r	Outcome
1T ($TESTS$)	114.5	0.38	0.16	Failed to reject H_0
1P ($PROD$)	90	0.82	0.04	Failed to reject H_0
1Q ($QLTY$)	81.5.5	0.53	0.11	Failed to reject H_0

As showed in the boxplots of Figure 3, only the $QLTY$ metric dataset still presents out-of-range values. In particular four datapoints for which $QLTY = 1.00$ and one for which $QLTY = 0.78$. The analysis of the standardized residual for the dataset (using both z-scores and modified z-scores) confirmed those as possible outliers. We inspected this datapoints and found no anomalies, in particular it seems that three, out of the four subjects who achieved a $QLTY$ score of 1.00, decided to focus on the quality of the delivered stories rather than delivering as much stories as possible, while one (who we might speculate having a higher skill level) delivered 10 stories (out of 13) with high quality. The trade-off between quality and productivity is discussed in Section VI. After the data reduction, the dataset contained 29 observations, 19 (6 pairs and 13 individuals) in the TF group and 10 in the TL group (3 pairs and 7 individuals).

3) Hypothesis Testing: Following the original study hypotheses, we first tested $1T$, $1Q$ and $1P$; and subsequently $2Q$ and $2P$. Finally we tested our additional hypotheses $2S_{2P}$ and $2S_{2Q}$. For $TESTS$, the medians of the two groups are slightly different, while there is a clear difference between the ranges between minimum and maximum values as well as 1st quantile and 3rd quantile. The $TESTS$ variable, as well as $PROD$ and $QLTY$, was not normally distributed according to the Lilliefors normality test [27], therefore we used a non-parametric test, Mann-Whitney U-test[3] [28] (as in the original experiment) to check for differences between the groups. As it could be anticipated from the boxplot, the null hypotheses in $1T$ failed to be rejected (W=114.5, p-value=0.38).

From the observation of the boxplot (Figure 3), it does not seem that there is any remarkable difference between TF and TL in terms of productivity. The U-test confirms that the null hypotheses for $1P$ failed to be rejected (W=90, p-value=0.82).

The same is true for $QLTY$, as it is visible from the third boxplot, the median values of the two groups are the same, the U-test confirms that also for $1Q$ the null hypotheses failed to be rejected (W=81.5, p-value=0.53). The effects size [29] for $1T$ is $r = 0.16$, for $1P$ $r = 0.04$ and for $1Q$ $r = 0.11$. All three effect sizes are to be considered *small* according to the guidelines provided by Kempenes et al. [30]. Table IV summarizes the results of hypotheses $1T, 1P$ and $1Q$.

To tackle hypotheses $2P$, we calculated the Spearman ranks correlation between $TESTS$ and $PROD$. The coefficient of correlation $\rho = 0.57$ is significant with p-value < 0.001. The relationship is analytically expressed by the following linear model:

$$PROD = 0.09 + 0.03 \times TESTS \tag{5}$$

The regression coefficient ($\beta = 0.03$) is significant (p-value$<$0.001) and indicates that there is an expected increase of 0.03 of productivity for every test written. Because in our

[3]Note that the p-values given are adjusted p-values due to the presence of ties in the dataset.

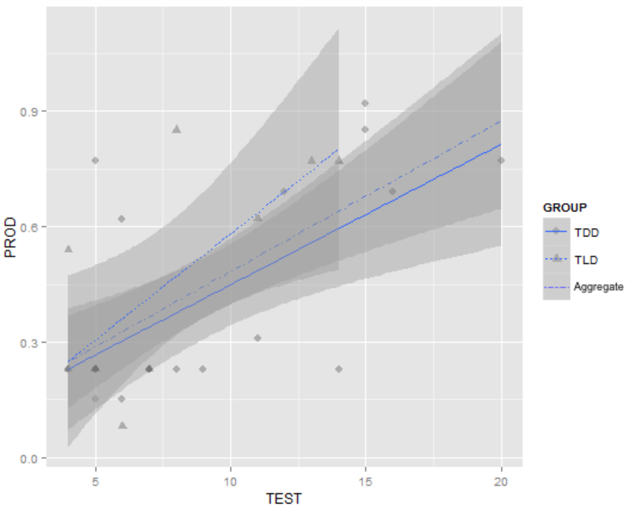

Fig. 4: Scatterplot and regression line for hypothesis $2P$. Please note that two instances (one TF, and one TL) of datapoint (4,0.23), five (3 TF, and 2 TL) of (5, 0.23), and three (2 TF, and 1 TL) of (7,0.23) are plotted.

context is not possible to have $TESTS = 0$, we do not give an interpretation of the intercept. The values of multiple R-squared ($R^2 = 0.40$) and residuals standard error ($\varepsilon = 0.21$) raise doubts about the reliability of this model. Multiple R-squared indicates that the number of tests accounts only for 40% of the variance in productivity, while the average error in predicting $PROD$ from $TESTS$ is 0.21. Hence, other hidden variables might have a remarkable impact on the outcome. We rejected the null hypothesis but addressed our concerns in Section VI. The same result remained consistent when analyzing the two experimental groups separately. Nevertheless, it is important to note that, despite TF development is foreseen to lead to a conspicuous number of unit tests and improved productivity, the correlation between those two variables is higher in the TL group. Figure 4 shows the relationships (overall, TF and TL) with 95% confidence intervals.

The same procedure is applied to test $2Q$. The correlation between $TEST$ and $QLTY$ ($\rho = 0.17$) is not significant (p-value=0.18). The correlation is expressed by the following inequality:

$$QLTY = 0.87 + 0.001 \times TESTS \tag{6}$$

Hypothesis $2Q$ failed to be rejected. At the same time, a non-significant correlation is observed between external quality and number of tests in TL ($\rho = 0.54, p - value = 0.06$), and TF ($\rho = -0.03, p - value = 0.54$) group. Figure 5 shows the correlations (overall, TF and TL) with 95% confidence intervals. A summary for the test of hypotheses $2P$ and $2Q$ is shown in Table V.

The analysis of covariance (ANCOVA) [31] was carried out to check the impact of the interaction between $TESTS$ and $SUBJ$ for both outcomes, $PROD$ and $QLTY$. Through the ANCOVA method we compared the slopes and intercepts of the two regression lines, one for each subject's type, assessing whether they present significant differences.

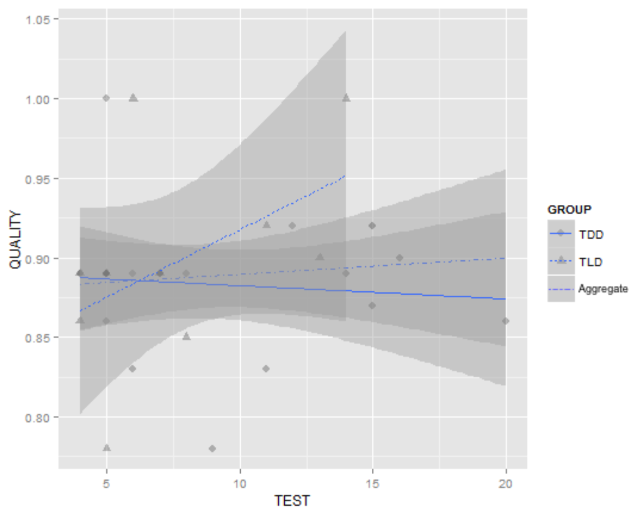

Fig. 5: Scatterplot and regression line for hypothesis $2Q$. Please note that two instances (one TF, and one TL) of datapoint (4,0.89), four (3 TF, and 1 TL) of (5, 0.89), and three (2 TF, and 1 TL) of (7, 0.89) are plotted.

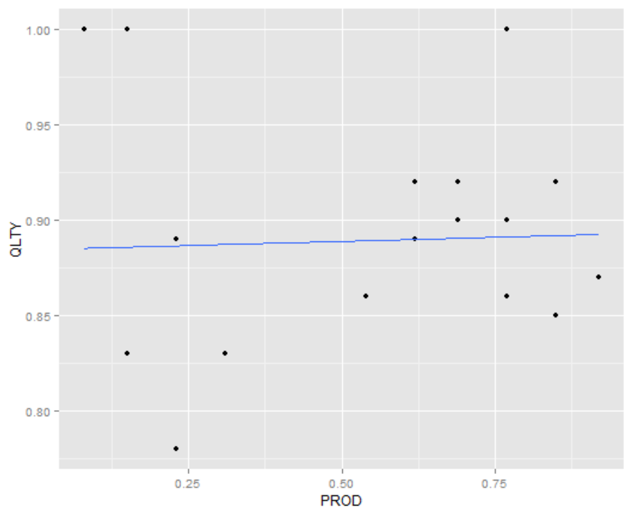

Fig. 6: Scatterplot for QLTY and PROD

TABLE V: Summary for test of hypotheses $2P$ and $2Q$, including subgroup analysis.

Hypotheses	ρ	p-value	β	R^2	ε	Outcome
2P	0.57	< 0.001	0.03	0.40	0.21	Reject H_0
(TL)	0.63	0.02	0.05	0.51	0.21	
(TF)	0.59	< 0.005	0.03	0.40	0.21	
2Q	0.17	0.18	0.001	0.008	0.05	Failed to reject H_0
(TL)	0.54	0.06	0.008	0.22	0.06	
(TF)	-0.03	0.54	-0.0008	0.008	0.04	

We tested differences between the slopes for hypotheses $2S_{2P}$ and did not found any significant result (p-value=0.79). Therefore, we failed to reject H_{0-2Sa} and tested the second sub-hypothesis, H_{0-2Sb}, checking for differences in the intercepts, by forcing a common slope between the two regression lines. In this case too, no statistically significant result was yield (p-value=0.66). Hence, $SUBJ$ did not have an interaction in the correlation between $TESTS$ and $PROD$. The test for hypothesis $2S_{2Q}$ yield not significant results. Hypotheses H_{0-2Sa} failed to be rejected (p-value=0.80) as well as H_{0-2Sb} (p-value=0.18). Hence, $SUBJ$ did not have an interaction in the correlation between $TESTS$ and $QLTY$.

4) Qualitative results: In order to have a qualitative validation of this quantitative results, once the study was completed, we asked the participants to voluntary fill in an anonymous post-questionnaire, we retained data from 48 of the initial subjects. The results showed that the TF subjects were more confident in the code they wrote, while neither group expressed a preponderant favor in the level of confidence they had when modifying their code. There is no favor as well for the perceived re-usability of the code authored in both groups. Finally, both groups perceived having a set of automated tests as valuable. The majority of TF developers also think that TF implies more effort, and they agreed that the reason lies in the difficulty of switching the mindset from test-last to test-first. Nevertheless, both groups acknowledge the value of TF in achieving higher test coverage and less defects.

5) Quality checking: Before interpreting our results, we ran a quality check of our data and models. Although TF is expected to support both quality and productivity at the same time, a trade-off between external quality and productivity might intuitively take place. Some subjects might want to deliver as much user stories as possible, neglecting the quality; while others might prefer to deliver few, high quality, user stories. Even though, the metrics used to gauge external quality and productivity might appear intuitively correlated, Figure 6 shows no significant (p-value=0.81) correlation in our dataset.

The model identified in Figure 4 supports a weak relationship between the number of tests and the developers' productivity. The same relationship holds in each of the two experimental groups. The confidence in the inferences about the regression parameters depends on the degree to which their statistical assumptions are met [32]. We assessed these assumptions using the guidelines developed by Peña & Slate [32] with the *gvlma* R library[4]. All the assumptions were evaluated as acceptable.

V. Threats to Validity

In this section we report the threats to validity following the guidelines by Wohlin et al. [25]. Considering that our replication involves theory testing, we prioritized the threats to validity in decreasing order according to Cook et al. [33].

A. Internal Validity

The internal threats to the validity of our study are mostly related to the *selection process*, since the selected subjects were not representative for the population of software developers; and the *post-experimental mortality*, since 38% of the artifacts delivered by the subjects were discarded according to the

[4]http://cran.r-project.org/web/packages/gvlma/index.html

quality criteria, as an indication of an overall low testing skill level. *Social threats to internal validity* might arise from the analysis of the results of the post-questionnaire. It appears that the subjects, in both groups, had a positive perception of the effects of TF, but at the same time agreed that the application of TF development takes time and effort. Therefore, the subjects receiving the less desirable treatment (i.e. TF) might have wanted to compensate for it by putting extra effort in tackling the task. On the other hand, the subjects might have been demotivated by being assigned to a treatment which is perceived as less desirable. In this regard, it is important to consider that the experimental hypotheses were disclosed to the subjects only after the completion of the experiment. Finally, the *process conformance* in the case of the TF treatment is another threat to the internal validity. The subjects were reminded about the importance of process conformance and were asked to *follow the rules* despite time pressure. When asked to assess their level of conformance, 87% of the subjects using TF responded 3 or more on a scale from 1 to 5.

B. Construct Validity

Our study design is subject to the same threats to construct validity as the original study. In particular the *mono-operation bias* due to the use of only one experimental task, and the *mono-method bias* due to the single method used to measure the subjects' productivity and the quality of the artifacts they produced. In this regard, the number of tests is not biased, being measured directly by a simple count. An improved metric that would also take into account the quality of tests represents material for future replications (See Section VII). Even though not formally controlled, some subjects were exposed to another *treatment*, i.e. pair programming. This poses a threat due to the *interaction of different treatments* that we controlled by testing two additional hypotheses. A *restricted generalizability across constructs* might affect our study, since there are other constructs (e.g. internal code quality, maintainability) that could be effected by the treatment but have not been measured. The subjects could have been exposed to the *Hawthorne effect* [34] by being observed and encouraged by the researchers during the experiment. On the other hand, no threat due to *evaluation apprehension* could have taken place since, for the purpose of the course, the subjects knew that their evaluation was based only on the previous lab sessions. Finally, the *experimenters expectancies* threat was limited by involving two researchers with no particular expectations on the outcome of the experiment.

C. Conclusion Validity

The ability of drawing correct conclusions from the results of our experiment might be hindered by the low statistical power of the test due to the small sample size. Nevertheless, the Mann-Whitney U-test is still robust when the sample size is approximately 20 [35]. We do not perceive *fishing* as a possible threat. When carrying out the data analysis one researcher acted as a supervising entity to limit the fostering for specific outcomes by who was performing the analysis. Since only one researcher performed the actual data analysis the error rate, i.e. significance level, was not adjusted. The *treatment implementation* was reliable, since benchmarked treatments were applied to all subjects at the same time. One possible threat is the *heterogeneity of the subjects*. Although we did not control such factor, heterogeneity should be reduced since the subjects had a similar background. Finally, no *external events* disturbed the execution of the experiment.

D. External Validity

The external validity of our study might be affected by different factors. The overall skill level and experience of the subjects might pose an *interaction of selection and treatment* threat to validity. Although, the studies by Müller and Höfer [36] and Philipp [37], show that the most skilled students could perform as good as experts, the subjects of this study were computer science students with no or little experience in TF development and limited software testing experience in general. The simplicity of the experimental task might pose a threat in the *interaction of setting and treatment*, although it should be taken into account that the limited time for completing the task can be representative of the time pressure in a real world setting. On the other hand, using the same experimental task of several previous studies — including the original study — eases the replication process as well as meta-analyses studies.

VI. Discussion and comparison of Results

In this section we compare the results of this experiment against the original, dividing them into consistent and different results. Furthermore, in the light of the changes in this replication and the results they yield, we discuss the current state of knowledge. It is worth noting that we decided to leave out a numerical comparison between the two experiments' measures for $TESTS$, $PROD$, $QLTY$ and the impact of the development technique on them due to the different metrics used in the two studies.

From the comparison of the outcome of hypotheses $1P$ and $1Q$, we found the same results as the original study. Our evidence, as well as the one provided in the original study, did not show any significant difference between TF and TL development in terms of productivity and quality. The original result is replicated in the correlation study between the number of tests and productivity, where a significant correlation was found; and between the number of tests and quality, where no correlation was found. Remarkably, we got a different result for hypotheses $1T$. While in the original study the TF group produced significantly more tests than the TL group, in our study no difference was found between the two groups.

We identified a discrepancy with the expected impact of the changes on the outcome. In particular, the use of pair programming impact neither productivity nor external quality. Hence, we can conclude that pair programming did not influence our results. Through this replication we contributed to increase confidence in the results obtained in the second stage of the original study. In particular the time pressure on the subjects and the use of pair programming did not impact the correlations already discovered in the original study. Although, the correlation between tests and productivity is significant, its effect size is small, leaving an open window to other factors. In particular, according to Maxwell and Deaney [38], the pre-existing differences between the subjects, have a major impact on the outcome. Since the number of tests explains only 40%

TABLE VI: Comparison of results between the original study and this replication

Hypothesis	Original	Replication
1T	Reject H_0	Failed to reject H_0
1P	Failed to reject H_0	Failed to reject H_0
1Q	Failed to reject H_0	Failed to reject H_0
2P	Reject H_0	Reject H_0
2Q	Failed to reject H_0	Failed to reject H_0

of the variability in productivity in our study and 60% in the original study, the differences in subjects' skills and experience might be an important predictor as well [39], [40]. At the same time we can argue that if the number of tests improves productivity when the subjects have little skill and experience it should have beneficial effects on more skilled subjects too.

The suggested impact of skill and expertise might be even more accentuated in the prediction of quality. In the first place, the variability in quality is high when few tests are written, while it tends to decrease when more tests are written (see Figure 5). Following our results, as well as the explanation given in the original study, the skill level could offset the low number of tests, allowing skilled subjects to still reach high level of quality. Arguably, when the effects of the pre-exisiting skill are not hindered by the new technique, as in the case of TF developers, the impact on quality is improved although it is not significant. The change in the experimental condition that forced us to include pair-programming did not impact the results. Therefore, we can conclude that the knowledge we generated is consistent despite the subjects' type (pairs vs. individuals).

Finally, our results did not fully reproduce the results of the first stage of the original study. In particular, the fact that we could not observe significant difference in the number of tests between TF and TL development is alarming, due to the central role that tests play in the TF approach. One explanation is that the a priori change in the time required to complete the task, and the subsequent pressure on the subjects applying the new technique might have lowered the number of tests they were able to write. On the other hand, the conformance to the process might have been lowered, although we took some countermeasures, i.e. by inspecting and removing unsuitable artifacts. Our study was not able to identify differences in productivity and quality despite the changed settings. Table VI compares the results of the original study and this replication.

VII. Conclusion and Future Work

With this study we replicated a controlled experiment by Erdogmus et al. (the original study) that reported a significant difference in the number of tests developed by a test-first group over a test-last group, whereas no difference was found in terms of productivity and external quality. The study also reported an analysis of correlation between the number of tests and productivity as well as external quality, that we also replicated. In the replication process some changes to the original context were necessary. We had a limited time available to complete the experimental task, therefore we had to simplify the task and change the metrics necessary to measure the variables; we were forced to introduce pair-programming and consequently checked for interaction effects by testing two additional hypotheses.

We were not able to find any differences between the TF and TL developers for any of the dependent variables (i.e. number of tests, productivity and external quality), contradicting the result of the original study, where a difference between the two groups was found regarding the number of tests. We conclude that TF does not improve nor worsen the external quality nor the productivity when compared to the TL approach, also when it is used along with pair-programming and under time pressure.

The results of our study about the relationship between the use of TF, productivity, and quality under different settings shows that other variables, hiding the postulated effects of TF, are yet to be found. The only significant correlation we found confirmed the results of the original study. In particular we expanded the context in which a higher number of tests appears to be correlated with a higher level of productivity. On the other hand we did not observe a correlation between tests and quality, which leaves room for other variables, e.g. the subject's skill and experience, to come into play.

To sum up the intentions for future works, we plan to carry out a family of replications in which each study will isolate a new variable, starting from the ones we could recognize in this study (i.e. subject's skill and experience), analyze their interactions, gauge their impact on tests, quality and productivity. Major improvements to this study are, for example, to have a bigger sample, as well as to use more precise metrics that would take into account the quality of the tests, and to have a formal assessment of TF process conformance using automated tools [41]. Moreover, experimentation in industrial context is needed in order to provide practitioners with recommendation on the use of TF development.

References

[1] K. Beck, *Test-driven Development: by Example*, ser. The Addison-Wesley signature series. Addison-Wesley, 2003.

[2] D. Astels, *Test Driven development: A Practical Guide*. Prentice Hall Professional Technical Reference, 2003.

[3] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus, "What Do We Know About Test-driven Development?" *Software, IEEE*, vol. 27, no. 6, pp. 16–19, 2010.

[4] M. Siniaalto and P. Abrahamsson, "Does test-driven development improve the program code? alarming results from a comparative case study," *Balancing Agility and Formalism in Software Engineering*, pp. 143–156, 2008.

[5] H. Erdogmus, M. Morisio, and M. Torchiano, "On the Effectiveness of the Test-First Approach to Programming," *IEEE Transactions on Software Engineering*, vol. 31, no. 3, pp. 226–237, 2005.

[6] N. Juristo and S. Vegas, "Using differences among replications of software engineering experiments to gain knowledge," in *Proceedings of the 2009 3rd International Symposium on Empirical Software Engineering and Measurement*, ser. ESEM '09. Washington, DC, USA: IEEE Computer Society, 2009, pp. 356–366. [Online]. Available: http://dx.doi.org/10.1109/ESEM.2009.5314236

[7] F. Shull, J. Carver, S. Vegas, and N. Juristo, "The Role of Replications in Empirical Software Engineering," *Empirical Software Engineering*, vol. 13, no. 2, pp. 211–218, 2008.

[8] B. Turhan, L. Layman, M. Diep, H. Erdogmus, and F. Shull, *How Effective Is Test Driven Development?* O'Reilly Media, 2010.

[9] Y. Rafique and V. Misic, "The effects of test-driven development on external quality and productivity: A meta-analysis," 2012.

[10] L. T. Xu S, “Evaluation of test-driven development: An academic case study,” *Studies in Computational Intelligence*, vol. 253, pp. 229–238, 2009.

[11] V. Bhadauria, “To Test Before Or To Test After-An Experimental Investigation Of The Impact Of Test Driven Development,” Ph.D. dissertation, The University of Texas at Arlington, 2009.

[12] S. Yenduri and L. Perkins, “Impact of using test-driven development: A case study,” *Software Engineering Research and Practice*, pp. 126–129, 2006.

[13] B. George, “Analysis and quantification of test driven development approach,” 2002.

[14] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. A. Visaggio, “Evaluating advantages of test driven development: a controlled experiment with professionals,” in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. ACM, 2006, pp. 364–371.

[15] B. George and L. Williams, “A structured experiment of test-driven development,” *Information and Software Technology*, vol. 46, no. 5, pp. 337–342, 2004.

[16] L. Madeyski, *Test-driven development: An empirical evaluation of agile practice*. Springer, 2009.

[17] L. Huang, “Analysis and Quantification of Test First Programming,” Ph.D. dissertation, The University of Sheffield, Aug. 2007.

[18] M. Pančur and M. Ciglarič, “Impact of test-driven development on productivity, code and tests: A controlled experiment,” *INFORMATION AND SOFTWARE TECHNOLOGY*, vol. 53, no. 6, pp. 557–573, Jun. 2011.

[19] E. M. Maximilien and L. Williams, “Assessing test-driven development at ibm,” in *Software Engineering, 2003. Proceedings. 25th International Conference on*. IEEE, 2003, pp. 564–569.

[20] T. Bhat and N. Nagappan, “Evaluating the efficacy of test-driven development: industrial case studies,” in *Proceedings of the 2006 ACM/IEEE international symposium on Empirical software engineering*. ACM, 2006, pp. 356–363.

[21] G. Melnik and F. Maurer, “A cross-program investigation of students’ perceptions of agile methods,” in *Software Engineering, 2005. ICSE 2005. Proceedings. 27th International Conference on*. IEEE, 2005, pp. 481–488.

[22] D. S. Janzen and H. Saiedian, “A leveled examination of test-driven development acceptance,” in *Software Engineering, 2007. ICSE 2007. 29th International Conference on*. IEEE, 2007, pp. 719–722.

[23] T. Flohr and T. Schneider, “Lessons learned from an xp experiment with students: Test-first needs more teachings,” *Product-Focused Software Process Improvement*, pp. 305–318, 2006.

[24] J. C. Carver, “Towards reporting guidelines for experimental replications: A proposal,” in *RESER2010: Proceedings of the 1st International Workshop on Replication in Empirical Software Engineering Research, Cape Town, South Africa*, vol. 4, 2010.

[25] C. Wohlin, *Experimentation in Software Engineering: an Introduction*. Springer, 2000, vol. 6.

[26] D. T. Sato, H. Corbucci, and M. V. Bravo, “Coding dojo: An environment for learning and sharing agile practices,” in *Agile, 2008. AGILE’08. Conference*. IEEE, 2008, pp. 459–464.

[27] H. W. Lilliefors, “On the kolmogorov-smirnov test for normality with mean and variance unknown,” *Journal of the American Statistical Association*, vol. 62, no. 318, pp. 399–402, 1967.

[28] H. B. Mann and D. R. Whitney, “On a test of whether one of two random variables is stochastically larger than the other,” *The annals of mathematical statistics*, vol. 18, no. 1, pp. 50–60, 1947.

[29] C. O. Fritz and J. J. Morris, Peter E.and Richler, “Effect size estimates: Current use, calculations, and interpretation,” *Journal of Experimental Psychology: General*, vol. 141, pp. 2–18, 2012.

[30] V. B. Kampenes, T. Dybå, J. E. Hannay, and D. I. Sjøberg, “A systematic review of effect size in software engineering experiments,” *Information and Software Technology*, vol. 49, no. 11, pp. 1073–1086, 2007.

[31] D. C. Howell, *Statistical methods for psychology*. Wadsworth Publishing Company, 2012.

[32] E. A. Pena and E. H. Slate, “Global validation of linear model assumptions,” *Journal of the American Statistical Association*, vol. 101, no. 473, pp. 341–354, 2006.

[33] T. D. Cook, D. T. Campbell, and A. Day, *Quasi-experimentation: Design & analysis issues for field settings*. Houghton Mifflin Boston, 1979.

[34] J. G. Adair, “The hawthorne effect: A reconsideration of the methodological artifact.” *Journal of Applied Psychology*, vol. 69, no. 2, p. 334, 1984.

[35] M. P. Fay and M. A. Proschan, “Wilcoxon-mann-whitney or t-test? on assumptions for hypothesis tests and multiple interpretations of decision rules,” *Statistics surveys*, vol. 4, p. 1, 2010.

[36] M. Müller and A. Höfer, “The Effect of Experience on the Test-driven Development Process,” *Empirical Software Engineering*, vol. 12, no. 6, pp. 593–615, 2007.

[37] M. Philipp, “Comparison Of The Test-Driven Development Processes Of Novice And Expert Programmer Pairs,” 2009.

[38] S. E. Maxwell and H. D. Delaney, *Designing experiments and analyzing data*, 2004.

[39] L. Madeyski, “The impact of test-first programming on branch coverage and mutation score indicator of unit tests: An experiment,” *Inf. Softw. Technol.*, vol. 52, no. 2, pp. 169–184, Feb. 2010. [Online]. Available: http://dx.doi.org/10.1016/j.infsof.2009.08.007

[40] A. Causevic, D. Sundmark, and S. Punnekkat, “Test case quality in test driven development: A study design and a pilot experiment,” in *Evaluation Assessment in Software Engineering (EASE 2012), 16th International Conference on*, 2012, pp. 223–227.

[41] P. Johnson and H. Kou, “Automated Recognition of Test-Driven Development with Zorro,” in *AGILE 2007*. IEEE, 2007, pp. 15–25.