Impact of process conformance on the effects of test-driven development

Conference Paper · September 2014

DOI: 10.1145/2652524.2652526

3 authors:

Davide Fucci
University of Hamburg
19 PUBLICATIONS 43 CITATIONS
SEE PROFILE

Burak Turhan
Brunel University London
148 PUBLICATIONS 1,358 CITATIONS
SEE PROFILE

Markku Oivo
University of Oulu
139 PUBLICATIONS 698 CITATIONS
SEE PROFILE

The Impact of Process Conformance on the Effects of Test-driven Development

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

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

Markku Oivo
Department of Information Processing Science
University of Oulu
Oulu, Finland
markku.oivo@oulu.fi

ABSTRACT

Background: One limitation of the empirical studies about test-driven development (TDD) is knowing whether the developers actually followed the advocated test-code-refactor cycle. Research dealt with the issue of process conformance only in terms of internal validity, while investigating the role of other confounding variables that might explain the controversial effects of TDD. None included process conformance as a fundamental part of the analysis.
Goal: We want to examine the impact of process conformance on the claimed effects of TDD on external quality, developers' productivity, test suite quality.
Method: We used the data collected during a previous study in order to create regression models in which the level of process conformance is used to predict external quality, productivity and tests thoroughness.
Result: Based on our analysis of the available data (n = 22), we observe that neither quality (p-value=0.21), productivity (p-value=0.80), number of tests (p-value=0.39) nor coverage (p-value = 0.09) can be regressed from the level of TDD process conformance.
Conclusion: Whilst based on a small sample, we raise concerns about the way developers interpret TDD. We also question whether the cost of strictly following TDD is going to pay-off in terms of external quality, productivity and tests thoroughness.

1. INTRODUCTION

Test-driven development (TDD) is a software development methodology that has been known since the beginning of 2000s as a central practice of extreme programming [51, 31, 16]. TDD strayed also outside the extreme programming movement due to its claimed positive effects on software quality (both internal and external), test quality and developers' productivity [37, 5, 17], until becoming part of the curricula for computer science and computer engineering degrees [12, 39]. On the other hand, TDD has been criticized for its difficult integration into the everyday's workflow, being counterintuitive and costly to implement [26, 7, 1].

TDD is based on the fast and short iteration of three phases [5]:

1. Create a unit-test for a functionality not yet implemented,
2. Develop the necessary code to make the test pass,
3. Refactor [15]

The need for upfront design is set aside and the unit-tests drive the design and implementation decision. Hence TDD should also be considered a design technique that produces a test suite as a byproduct, rather then a testing technique [11].

The need to validate the claims about TDD prompted the software engineering research community to conduct several empirical studies that in most cases resulted in controversial results [44, 46]. One common argument reported in such studies is that a model to understand and benchmark the effects of TDD should include other variables, e.g. the skills and experience, whether TDD is accompanied by another development methodology. Some proposed TDD to be accompanied by other agile techniques, like pair-programming [55, 32]; others focused on the skills of the TDD developers [8, 40], while further studies investigated the central component of TDD, i.e. unit-tests [13, 20].

A common theme in the existing evidence about the effects of TDD is that process conformance, i.e. the extent to which the actual TDD cycle is followed, is considered marginally and mostly reported as a threat to the internal validity. In this study we propose a model (Fig. 1) in which the relation between conformance to TDD and its effects is formally investigated. Hence, the novelty of this study is being the first to formally investigate the relation between conformance to TDD and its postulated effects.

The paper is organized as follows: Section 2 presents a review of the existing literature about process conformance studies in software engineering, and of particular interest for TDD research. Section 3 introduces the study context, the metrics used, the research questions and hypotheses. Section 4 reports the analysis of the data and the results of the study. The threats to validity are examined in Section 5, while the results are discussed in Section 6. Finally, Section 7 presents the conclusion and addresses future work.

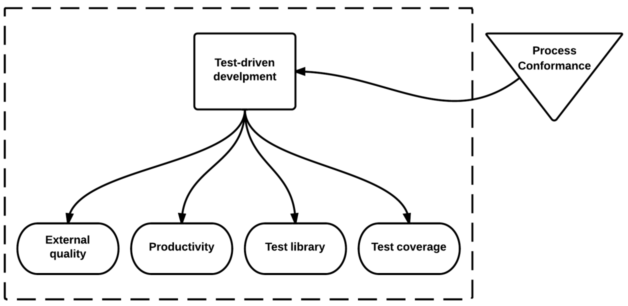


Figure 1: Impact of process conformance on TDD

2. RELATED WORK

In this section we survey process conformance studies in software engineering, and then examine the ones addressing the subject specifically for test-driven development.

2.1 Studies About Process Conformance in SE

Sørumgård defines process conformance in the context of software development, informally as "the degree of agreement between the software development process that is really carried out and the process that is believed to be carried out" [48]. The author aimed to uncover the importance of process conformance in software engineering field. The study helped to uncover four key components to measure process conformance in software development: model for definition, measurement based on the model, an alternative measurement and guidelines for modification of a process [48].

The study by Cook and Jonathan [10], provides deeper insight in the relationship between a formal model and its execution for high lever processes in software development. In particular they realized that if the the model and the executions diverges, then something crucial is happening. They proposed several metrics to aid the validation of a process, along with the type of data to collect or rather ignore. These metrics were demonstrated using the Test Unit task from [29].

The lack of adequate process conformance measurements in software engineering is pointed out by Silva and Travassos [47]. In particular, the authors are concerned about the obtrusiveness of the existing methods to gauge conformity. They proposed and validated a tool in industrial settings to support the adherence to the prospective-based reading requirements elicitation methodology. The use of the tool to keep the subjects conforming to the process resulted in a significant reduction in the time needed for elicitation [47]. Notably, the authors reflect about the importance of measuring process conformance in order to have a better understanding of the process itself.

Zazworka et al. [53] developed a new approach to identify disconformities in software development lifecycle processes. They defined a template for the completition of a process and looked for patterns of non conformity in the collected data. Further, they used data visualization techniques to present and evaluate the risk associated with non conformity behaviors. The authors claim that, with this approach, they can find the deviations from the model already from the start and throughout the whole process [53]. The method presented in Zazworka et al. has been subsequently tested to gauge process conformance during an XP course, and identified process conformance violations as a major hinder in instructing developers about new processes [54].

2.2 Studies About Process Conformance for TDD

Different studies tackled the problem of quantify process conformance, specifically for TDD. In particular, Wang and Erdogmus [50] measures process with the goal of improving it by using micro-level analysis of the data extracted from the developers' IDE. They developed a tool that mines Hackystat [1] data looking for patterns that are typical of the TDD cycle (i.e. lines of code to lines of test ratio, testing effort vs. production effort and timestamp).

Madeyski and Szala [33] conducted an experiment about TDD effectiveness. In order to improve the internal validity of their study they measured process conformance. Their approach, similar to the previous, relied on the duration of development events that were subsequently categorized into passive and active. They were able to infer TDD process conformance from the passive-to-active time ratio, concluding that TDD subjects attain higher active time than test-last developers.

Also the studies by Johnson and Kou [27, 30], use low level development episodes to calculate process conformance. Through several heuristics, they were able to identify 22 different events that usually take place during the development flow (e.g. adding a new class, compiling, executing the test suite), and categorize them into 8 coarse-grained class. Each sequence of episodes is then marked as TDD compliant or not using a rules-based system,. They implemented a tool, Zorro, and validated it by comparing the result of automated analysis to the manual analysis of a video recording of the developers activity. Zorro classified correctly 89% of the development episodes in the first run of the experiment in university settings and 85% during the second run in industrial settings. Pedroso [41], presents the result of the experimentation of an enhanced version of Zorro, Besouro, which is 8% more precise.

3. STUDY DESIGN

With this study we want to understand the relationship between process conformance and the the postulated benefits of TDD, specifically improved external quality, improved productivity, enhanced number of unit-tests written and improved coverage [37, 4, 18, 19].

3.1 Research Questions

The main question this study tries to answer is:
What is the relationship between the conformance to TDD and its observed effects?
In particular we want to investigate the impact on external software quality, developers' productivity and tests suite, in terms of number of tests produced as well as the thoroughness of tests. To answer the research we constructed a regression model (Sec. 4.3) for each of the variables under study. We then took the same approach, but considering two subgroups, low and high conformant subjects.

[1] A framework for collection and analysis of software development process data — https://code.google.com/p/hackystat

Table 1: Summary of the study context variables

Variable	
Subjects (Recruited / Observed / Retained)	33 grad., 25 undergrad. / 27 / 22
Subject Unit (Observed / Retained)	20 individuals, 7 pairs / 15 individuals, 7 pairs
Subject Type	Mix of undergraduate and graduate students
Programming Environment	Java, Eclipse, JUnit
Programming Task	Robert Martin's Bowling Scorekeeper
Programming Task Type	Fine grained with incremental difficulty
Task's user stories	13
Time to Complete the Task	One lab session $\sim$ 3 hours

3.2 Study Context

The study reported in this paper is based on our controlled experiment about the effects of TDD published elsewhere [21]. In the study, the recruited subjects were randomly divided into two groups (non-TDD and TDD) for experimental purposes. The goal of the experiment was to check whether significant differences exists between TDD developers and test-last developers, in terms of external quality and productivity. Moreover, we investigated the correlation between the number of the tests developed and the external quality and productivity. In this paper we consider the data gathered only from the TDD group to understand how the level of conformance of the subjects to the prescribed TDD process impacts external quality, productivity and test quality.

The software artifacts used in this study were produced by senior undergraduate and junior graduate subjects in the context of an academic course about software testing organized by the department of Information Processing Science at the University of Oulu, Finland in the Fall term of 2012. The subjects received 20 hours of hands-on training about unit-testing and TDD. During the last session the subjects were asked to implement the Bowling ScoreKeeper exercise used for previous experiments [20, 13, 22] in a TDD fashion. The session duration was 3 hours and we retained data from 27 subject units, 20 single and 7 pairs. In our previous study we assessed that there is no difference between pairs and single developers [21].

It should be noted that, for the controlled experiment encompassing this study, we gathered data about the subjects (including the ones in the non-TDD group) skills and experience using questionnaires. There were 41 novices and 15 professionals with experience varying between 6 months and 10 years, with average of 2.6 years [21]. Although this information can not be directly used for the purpose of our study, it gives an idea of sample in terms of programming experience. Table 1 summarizes the context in which the study took place.

3.3 Data Collection

Before the experimental task and data collection phases, the subjects agreed to participate to the study, although the hypotheses were not disclosed to them. Along with the list of user stories to implement, the subjects were given a template project (30 LOC) containing the API and naming conventions to be followed. Once the eperiment was over, we collected the artifacts developed by all the subjects (production and test code) and proceeded to examine them. We removed 5 artifacts which were either empty or impossible to compile, leaving us with 22 artifacts. The mean LOC value was 186.27 (sd=101.82), the mean LOC for production code was 82.59 (sd=39.61) while for test code 103.68 (sd=81.9).

3.4 Metrics

To measure quality we use a defect-based metric calculated as:

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

where $\#d.u.s.$ is the number of delivered user stories. A user story is considered delivered when at least 50% of the acceptance tests associated with that user story pass. Whereas, the quality of the single user story is:

$$QLTY_i = \frac{\#assert_{(T)}}{\#assert} \tag{2}$$

In other words, it represents the ratio of the passing assert statements from the acceptance test associated with that user story over all the assert statements associated with the same user story. Our test suite included 56 acceptance tests for the 13 user stories that constituted the task. $QLTY$ is a ratio variable that can vary in the interval [0, 1.0].

$PROD$ is also calculated based on user story. In particular:

$$PROD = \frac{\#d.u.s.}{\#u.s.} \tag{3}$$

which represent the ratio of delivered user stories. It is important to note that equation 3 does not include a measure of the time necessary for each subject to complete the task since the time was the same for all subjects (3 hours). $PROD$ is a ratio variable that can vary in the interval [0, 1.0].

The variable $TEST$ was calculated by simply counting the tests present in test suite developed by the subjects. $TEST$ is a ratio variable that can vary in the interval [1, ∞].

Along with $TEST$, a second measure that gauges the quality of the test suite developed by the subject is $TCOV$. It represents the percentage of code covered by the tests [36]. Although several ways to calculate code coverage exists [28], we adopted block coverage [25] as it was implemented by the tool[2] used during for the study. $TCOV$ is a ratio variable with interval [1, 100]. The level of conformance was calculated using the output of the Besouro [3] tool. In particular:

$$CONF = \frac{\#episodes_{(TDD)}}{\#episodes} \times 100 \tag{4}$$

Conformance is then the percentage of the developers' episodes recognized as TDD compliant over all the episodes. $CONF$ is a ratio variable, that can be vary in the interval [0, 100].

[2]http://www.eclemma.org

[3]https://github.com/brunopedroso/besouro

Table 2: Formal Hypotheses

Name	H_0	H_1
H_{QLTY}	$QLTY = \beta \times CONF + \alpha, \beta = 0$	$QLTY = \beta \times CONF + \alpha, \beta \neq 0$
$H_{QLTY(HC)}$	$QLTY(HC) = \beta \times CONF(HC) + \alpha, \beta = 0$	$QLTY(HC) = \beta \times CONF(HC) + \alpha, \beta \neq 0$
$H_{QLTY(LC)}$	$QLTY(LC) = \beta \times CONF(LC) + \alpha, \beta = 0$	$QLTY(HC) = \beta \times CONF(HC) + \alpha, \beta \neq 0$
H_{PROD}	$PROD = \beta \times CONF + \alpha, \beta = 0$	$PROD = \beta \times CONF + \alpha, \beta = 0$
$H_{PROD(HC)}$	$PROD(HC) = \beta \times CONF(HC) + \alpha, \beta = 0$	$PROD(HC) = \beta \times CONF(HC) + \alpha, \beta \neq 0$
$H_{PROD(LC)}$	$PROD(LC) = \beta \times CONF(LC) + \alpha, \beta = 0$	$PROD(LC) = \beta \times CONF(LC) + \alpha, \beta \neq 0$
H_{TEST}	$TEST = \beta \times CONF + \alpha, \beta = 0$	$TEST = \beta \times CONF + \alpha, \beta = 0$
$H_{TEST(HC)}$	$TEST(HC) = \beta \times CONF(HC) + \alpha, \beta = 0$	$TEST(HC) = \beta \times CONF(HC) + \alpha, \beta \neq 0$
$H_{TEST(LC)}$	$TEST(LC) = \beta \times CONF(LC) + \alpha, \beta = 0$	$TEST(LC) = \beta \times CONF(LC) + \alpha, \beta \neq 0$
H_{TCOV}	$TCOV = \beta \times CONF + \alpha, \beta = 0$	$TCOV = \beta \times CONF + \alpha, \beta = 0$
$H_{TCOV(HC)}$	$TCOV(HC) = \beta \times CONF(HC) + \alpha, \beta = 0$	$TCOV(HC) = \beta \times CONF(HC) + \alpha, \beta \neq 0$
$H_{TCOV(LC)}$	$TCOV(LC) = \beta \times CONF(LC) + \alpha, \beta = 0$	$TCOV(LC) = \beta \times CONF(LC) + \alpha, \beta \neq 0$

3.5 Conformance criteria

Part of our hypotheses are built on the assertion that the subjects could be splitted according to their conformance to TDD ($CONF$), i.e. low conformant (LC) and high conformant (HC), and that the two groups are statistically different. We looked at the descriptive statistics for the variable $CONF$, n=22, mean=78.09, sd=21.65, median=85, and the distribution of the data to select the median (85%) as a cut-off value to split the dataset according the level of conformance. In order to formally test for differences between the two obtained groups we used a non-parametric test, since the assumptions of homogeneity of variance (Levene's test [23] $p < 0.05$) and normality (Shapiro-Wilks's test [49] $p > 0.05$) are both violated.

The one-sided Mann-Whitney's test [14] ($CONF(HC) > CONF(LC)$) was significant ($p << 0.05$). Hence we conclude that, as expected, using the median value to split the dataset into two groups, there is a statistically significant difference between them in terms of process conformance.

The HC group contains 12 data points, while the LC group 10. Having established the two groups of subjects allows us to formally express the research hypotheses.

3.6 Hypotheses

The research question presented in Section 3.1 is formalized in research hypotheses presented in Table 2. For the regression study $CONF$ is the independent variable (IV), while $QLTY$, $PROD$, $TEST$ and $TCOV$ are the dependent variables (DV). Through the test of the hypotheses we want to identify the strength and the direction of the relationship between the IV and each of the DV. The same analysis is carried out considering the two groups, HC and LC, separately.

3.7 Research Method

The main methodology used to test the hypotheses and answer the research questions of this study is regression analysis [45]. Since the regression model presented in some of the hypotheses operates on a subset of our data, we first decided the criteria to split the dataset, then proceed to describe it, analyze possible outliers and correlations between variables. Finally, we build the models necessary to answer the research questions. All the analyses were carried out using R version 2.15.2.[4]. The significance level to test all the hypotheses is 0.05.

[4]http://www.r-project.org/

4. RESULTS

This section reports results of the data analysis. In particular the description of the dataset, the assessment of correlation between variables, and finally the actual regression models.

4.1 Descriptive Statistics

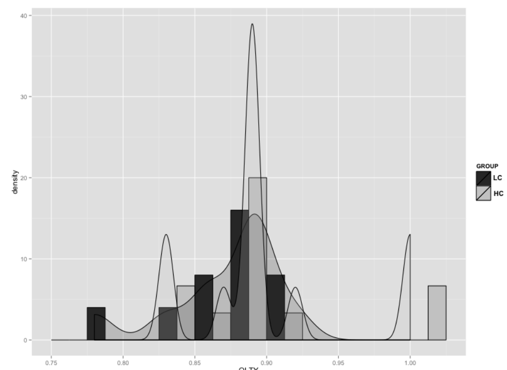
The data was first analyzed by descriptive analysis. In particular, the descriptive statistics for the variables $QLTY$, $PROD$, $TEST$, $TCOV$, $CONF$ and their subgroups are summarized in Table 3. Observing the means for the two

Table 3: Descriptive statistics for PROD, QLTY, TEST, TCOV and sub-groups

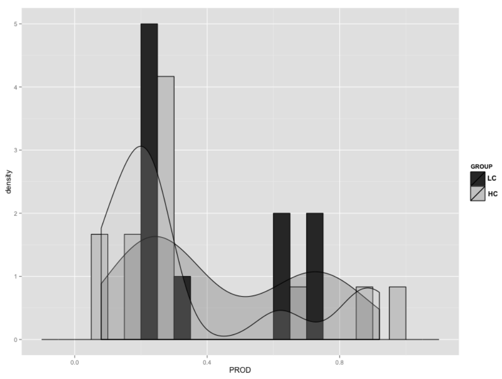
Variable	Mean	Std.Dev.	Min	Median	Max
$QLTY$	0.89	0.05	0.78	0.89	1.0
$QLTY(HC)$	0.89	0.054	0.83	0.89	1.0
$QLTY(LC)$	0.87	0.041	0.78	0.89	0.92
$PROD$	0.38	0.27	0.08	0.23	0.92
$PROD(HC)$	0.33	0.29	0.08	0.23	0.92
$PROD(LC)$	0.43	0.25	0.23	0.27	0.77
$TEST$	7.95	5.36	1.0	6.5	20.0
$TEST(HC)$	7.95	5.36	1	6.5	20
$TEST(LC)$	9.3	5.46	4	8	20
$TCOV$	81.25	18.83	32.1	88.15	99.5
$TCOV(HC)$	81.25	18.83	32.1	88.15	99.5
$TCOV(LC)$	87.98	11.24	66.7	90.4	99.2

groups regarding $QLTY$ and $PROD$, the impression is that more conformant subjects are slightly less productive although the quality that they achieved is very close, although slightly in favor of the high conformant group. It is important to note that, for the same two outcomes, the standard deviation values are rather large if compared to the means. Standard deviations are larger for HC groups for the $QLTY$, $PROD$ and $TEST$ variables, indicating that the data points are distant from the means and reflecting greater uncertainty for the subjects in this group. Against the expectation, $TEST$ is greater for the the less conformant subjects, hence also the mean value for $TCOV$ is increased for the LC group.

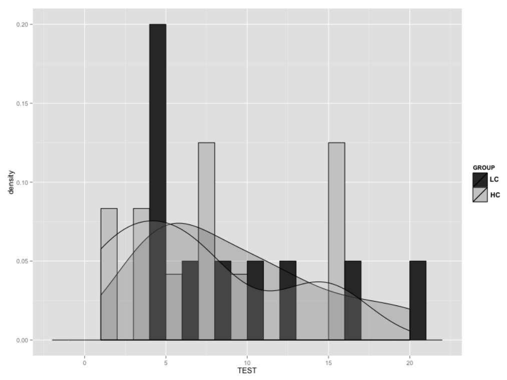
The histograms in Fig. 2 show the density distributions of the variables under study as well as the estimated density



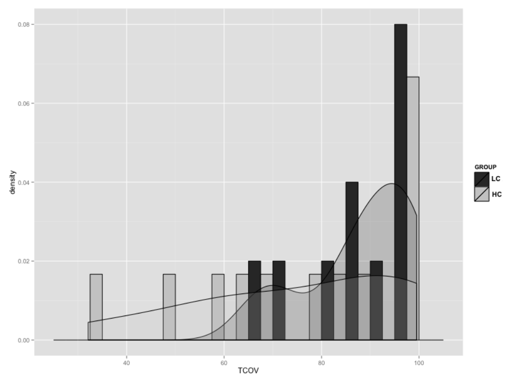
(a) QLTY



(b) PROD



(c) TEST



(d) TCOV

Figure 2: Density distribution for $QLTY$, $PROD$, $TEST$ and $TCOV$.

functions for both conformance groups. The distribution for the HC (2a) appears to be multi-modal, this might be a clue about the presence of other sources of variation (e.g. subjects' skills, experience), although the small sample size does not allow us to make any sound claim. Moreover, the isolated bar on the right-hand side of the plot indicates the possible presence of outliers. Also the distribution of the $PROD$ variable (2b) is multi-modal for both groups, with the values condensed on the left indicating low productivity for a substantial part of the subjects, and extreme values candidate to be outliers for the HC group.

For both groups, the distributions of $TEST$ is slightly skewed left, pointing that a large part of the subjects wrote few tests, which might be connected with the overall low level of productivity. Moreover, the inspection of Fig. 2c visually confirms that LC subjects tended to write more tests than HC subjects.

On the other hand, the distribution of the $TCOV$ variable (Fig. 2d) is skewed on the right for both groups. This might be again related to the fact that the few subjects who wrote a high number of tests were able to achieve high coverage, while the majority of subjects who wrote few tests achieved low coverage. The plot confirms that the baseline coverage of the subjects in the LC group is higher than their counterparts in HC. The plot also show some value, on the left hand side that are candidate to be outliers. At this stage we left out the analysis of the outliers, and considered for diagnose the outcome of the regression analysis 4.3.

4.2 Correlation Analysis

In order to probe into unusual interaction among the variables under study, we run a correlation analysis on the data set as a sanity check. We tested the assumptions for *Pearson's r* and *Spearman's ρ* coefficients to decide which measure of correlation to adopt. Both measures assume that the variables are at least interval and, in the case of Pearson's r, normally distributed. We used Shapiro-Wilk test of normality for all the variable and relative subgroups. According to the result of the test only $TCOV$ is normally distributed (W = 0.86, p-value = 0.0072). Hence, we used *Spearman ρ* to calculate all the correlations. The correlation coefficients, along with their significance, are reported in Table 4.

Table 4: Results for correlation measure.

	$QLTY$	$PROD$	$TEST$	$TCOV$
$QLTY$				
$PROD$	-0.09			
$TEST$	-0.13	0.69***		
$TCOV$	0.01	0.53*	0.72***	
$CONF$	0.28	-0.06	-0.19	-0.36

* $p < 0.05$
*** $p < 0.001$

The results indicate that there is a statistically significant positive correlation between the number of tests and the developers' productivity, as found in previous experiments [13, 20]. There is also a significant positive correlation between the number of tests and the test coverage, which is expected as having more tests increases the chances of covering different parts of a system [35, 34].

There is a weak correlation between test coverage and productivity, which follows from one of the claim of the TDD proponents: having a solid regression test suite allows the developers to add new features without the fear of breaking what has been previously implemented [5, 3]. Finally, none of the variables is significantly correlated with the independent variable $CONF$.

4.3 Regression analysis

Most of the previous studies on TDD used process conformance as a mean to check for internal validity [33, 30, 50]. In this section we analyze the data looking for analytical relationships between TDD process conformance and its claimed effects.

Having only one IV ($CONF$), our model consists of a simple linear regression. We want to check whether $CONF$ has a linear relationship with $QLTY$, $PROD$, $TEST$ and $TCOV$. Using the data we collected, we used ordinary least square (OLS)[9] in order to create model to test our hypotheses. Table 5 summarizes the results of our analysis. None of the models are significant, although it is useful to check the distribution of the data points and the shape of the regression lines in order to understand the phenomenon.

The statistical assumptions beneath the OLS have been evaluated as acceptable (p-value > 0.05) using *gvlma* method [42]. After assessing the models, we screened them for unusual observations like outliers and leverage points. The information about such points are combined in Figure 4.

Our perception is that those outlier observations are associated to very skilled subjects, maybe professionals, who are not representative of the sample. We rebuilt the models using the dataset without the outliers.None yielded any statistically significant coefficient. Table 6 summarizes the results of the hypotheses test.

Table 6: Summary of hypotheses testing

Hypothesis	**Outcome**
H_{QLTY}	Failed to reject H_0
$H_{QLTY(HC)}$	Failed to reject H_0
$H_{QLTY(LC)}$	Failed to reject H_0
H_{PROD}	Failed to reject H_0
$H_{PROD(hC)}$	Failed to reject H_0
$H_{PROD(LC)}$	Failed to reject H_0
H_{TEST}	Failed to reject H_0
$H_{TEST(HC)}$	Failed to reject H_0
$H_{TEST(LC)}$	Failed to reject H_0
H_{TCOV}	Failed to reject H_0
$H_{TCOV(HC)}$	Failed to reject H_0
$H_{TCVO(LC)}$	Failed to reject H_0

5. THREATS TO VALIDITY

The threats to validity analyzed in this section follow the guidelines proposed by Wohlin et al [52] and share mostly the same limitations as in our previous study [21].

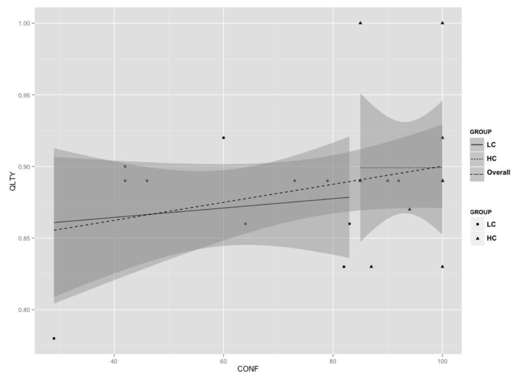
The main threat to *interal validity* of our study is the selection process. In fact, the subjects were not representative of the population of developers. Although our results showed not significant regression models, there might be other hidden variables that have a correlation with, or even cause, the observed effects. We do not foresee it as critical due to our experience in running other studies under similar settings [20, 21]. Since the tool we used is able to recognize process conformance to TDD up to 97% of the cases, we consider process conformance a minor threat to validity.

Regarding *construct validity*, the main threat is mono-operation bias since the constructs are studied using only one task, and the mono-method bias due to the single metric used to measure the variables. A threat due to interaction with a different treatment, namely pair-programming, might have taken place since — due to logistics limitations — some subjects worked in pairs. Nevertheless, in the controlled experiment encompassing this study [21] no confounding effects were found analyzing the same data. The side effects of process conformance might not be visible, i.e. there are other constructs (e.g. internal code quality) that are influenced or have an influence on the independent variable and were not observed or measured. Regarding social threats, we do not foresee hypothesis guessing as influential since the subjects were not aware of participating in the study until its conclusion. The evaluation apprehension, human tendency of being afraid of being evaluated [24], should not be critical since the subjects knew that their performance during the task were not used for the purpose of the course evaluation. Whereas, the Hawthorne effect [2] might have take place since the subjects were all the time observed by the researchers. Finally, we as researchers did not have any particular expectation over the results that might have biased the study.

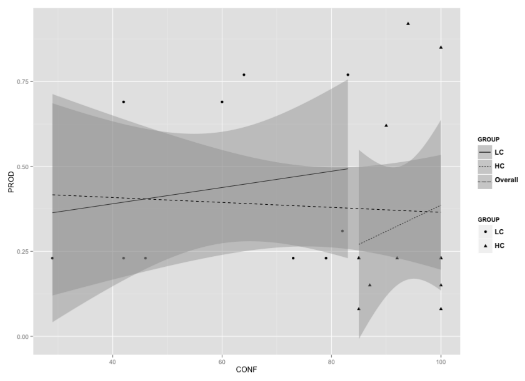
The main *conclusion validity* treat to our study lies in the low statistical power that our analysis could achieve due to the limited sample size [38], although this is balanced by using only one estimator per model. One researcher analyzed the data while the others acted like supervisors in order to limit fishing for specific outcome, this in turn implies that the error rate did not need to be adjusted.

The measures should be reliable since they were taken using automatic instruments — largely validated by previous studies — without involving human judgment and subjectivity. The implementation of the single treatment was reliable, since all the subjects developed the task at the same time using the same development approach. The subjects under study are supposed to be homogeneous by construct, being all selected from the same population (i.e. computer science students), nevertheless the analysis of the results showed outliers that could be explained by subjects coming from a different population (i.e. professional developers). Although, we cannot provide formal evidence, such phenomena poses a random heterogeneity of subjects threat to validity.

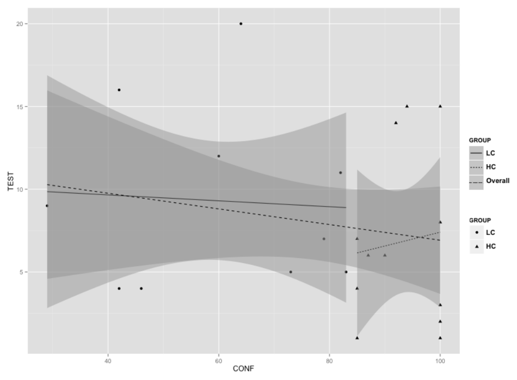
The interaction of selection and treatment as an external validity threat applies to our study. The subjects were students with little experience in programming and testing and even less experience it test-driven development. Although it might be difficult to generalize our findings to a population of professional developers and software engineers, we need to take into account that other studies [40, 43, 6] showed how the more skilled student can perform at the same level, or even outperform, more experience professionals. Finally, the simple task might not be representative of a real-world scenario — while we tried to make it more realistic by lim-



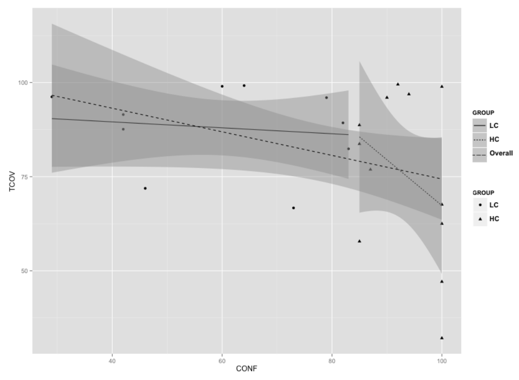
(a) CONF vs. QLTY



(b) CONF vs. PROD



(c) CONF vs. TEST



(d) CONF vs. TCOV

Figure 3: Scatter plot and linear regression lines for $QLTY$, $PROD$, $TEST$ and $TCOV$ for HC and LC groups. Confidence interval 0.95

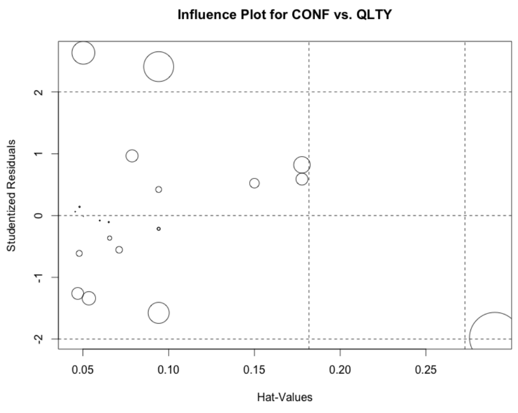
Table 5: Coefficient, significance level and R-squared for the regression models. *DF=20 for the overall dataset, DF=8 for the LC group, DF=10 for the HC group*

Model	β	t-test	p-**value**	R^2 **(effect size)**	**F-test**
$QLTY$	6.1e-4	1.28	0.21	0.076	1.65
$QLTY(HC)$	3.4e-6	0.001	0.99	1.8e-7	1.81e-6
$QLTY(LC)$	3.2e-4	0.44	0.66	0.24	0.19
$PROD$	-7.2e-04	-0.25	0.8	3.28e-3	0.06
$PROD(HC)$	0.007	0.56	0.58	0.03	0.31
$PROD(LC)$	0.002	0.52	0.61	0.03	0.27
$TEST$	-0.04	-0.87	0.39	0.036	0.75
$TEST(HC)$	0.08	0.33	0.74	0.01	0.11
$TEST(LC)$	-0.01	-0.17	0.86	0.003	0.03
$TCOV$	-0.31	-1.72	0.09	0.12	2.98
$TCOV(HC)$	-1.21	-1.22	0.24	0.13	1.49
$TCOV(LC)$	-0.07	-0.38	0.70	0.01	0.14

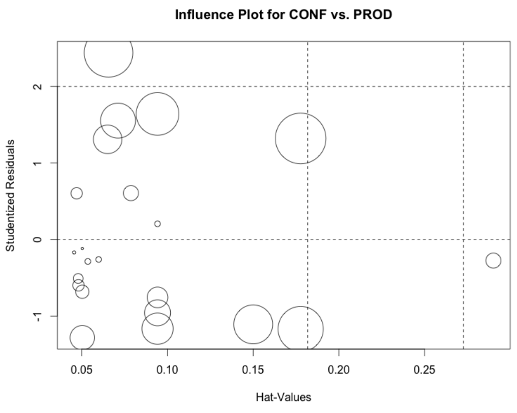
iting the time, hence increase the pressure on the subjects — posing an interaction of setting and treatment threat to validity.

6. DISCUSSION

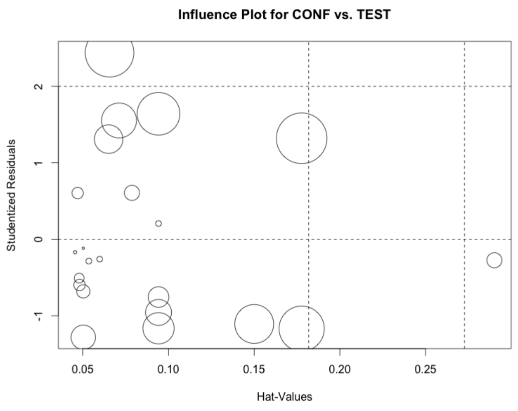
The trend of the regression lines for $QLTY$ and $PROD$ (Fig. 3a and Fig. 3b) indicate that HC subjects delivered artifacts with an higher baseline for quality at the expense of productivity. Such subjects chose to use their time to implement the user stories with high quality, but they did not consider the thoroughness of the tests (Fig.3d). We can speculate that part of the subjects, being novices, tended to write a lot of tests (Fig. 3c), hence improving coverage but, by focusing on the test-first feature of TDD and without taking into account other features like refactoring, resulted to be low conformant. On the other hand, the HC subjects show an improvement in terms of $TEST$ even if with a lower baseline. The subjects in the LC group might have deviated from the TDD process in order to deliver more user stories, hence the high baseline productivity (Fig.3b) with a lower quality if compared with the high conformant (Fig. 3a).



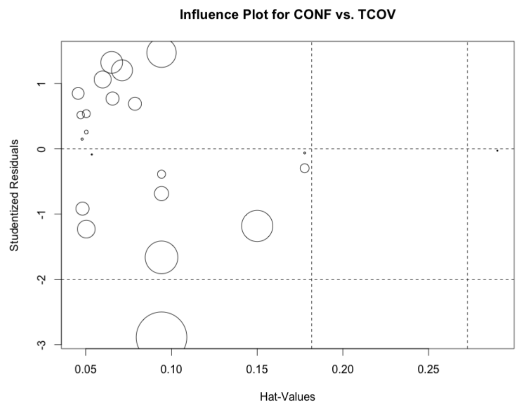
(a) CONF vs. QLTY



(b) CONF vs. PROD



(c) CONF vs. TEST



(d) CONF vs. TCOV

Figure 4: Influence plot. Data points above +2 or below -2 on the vertical y-axis are considered outliers. Above 0.2 on the x-axis lie points with high leverage. The size of the circle is proportional to its influence on the regression model.

Since their attention to focus on quality was preceded with getting things done, the quality of their code and tests are uniform throughout as opposed to the *HC* subjects.

Furthermore, even though based on a small sample of nonprofessionals, the results of this study are questioning whether the effort of introducing and enforcing the TDD practice among developers is truly beneficial. Our consideration is that, although at the beginning the effects of high conformance to TDD are not visible or hidden by other variables, they may manifest on the long run.

7. CONCLUSION

In this study we considered the pivotal role of process conformance when investigating the effects of TDD. Using data gathered in a previous study with university students as subjects, we devised regression models in which conformance to TDD is used to predict external quality, developers' productivity, number of developed tests and code coverage.

The data did not show any significant regression coefficient that could be used to create an effective linear model. We came across the same results after dividing the dataset into two sub-groups of high and low conformant subjects, and analyzing them separately.

Although we did not observe any significant relationship between the conformance to TDD and its claimed effects, our interpretation of the regression models can shed some light on the way developers approach the TDD cycle. It appears that the main concern of the subjects applying TDD is to write a fair amount of tests — whether before of after the actual production code — while omitting, for example, the refactoring phase. At the same time we argue that the effort put in implementing the TDD practice in the developers' workflow does not yield a viable return on the investment, at least in the short term.

Therefore, the intensions for future replications of this study are twofold: we want to investigate more the actual impact of TDD as a whole, and not only as a practice that reinforces the writing of tests before implementing a feature, while leaving out the prescribed refactoring phase. We want to involve industrial partners in order to understand the effects of TDD in the long run. We will take into account and control other variables that might inhibit the observation of the claimed TDD effects, e.g. pre-existing subjects' skill and experience, the use of other development techniques along with TDD, as well as increase the sample size in order to improve the design of the study.

Acknowledgment

8. REFERENCES

[1] P. Abrahamsson, A. Hanhineva, and J. Jäälinoja. Improving business agility through technical solutions: A case study on test-driven development in mobile software development. In *Business Agility and Information Technology Diffusion*, pages 227–243. Springer, 2005.

[2] J. G. Adair. The hawthorne effect: A reconsideration of the methodological artifact. *Journal of Applied Psychology*, 69(2):334, 1984.

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

[4] D. Astels. A new look at test-driven development. *Divulgado em http://blog. daveastels. com/files/BDD_Intro. pdf, acessado em*, 13(06):2010, 2006.

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

[6] J. Bowyer and J. Hughes. Assessing undergraduate experience of continuous integration and test-driven development. In *Proceedings of the 28th international conference on Software engineering*, pages 691–694. ACM, 2006.

[7] A. Causevic, D. Sundmark, and S. Punnekkat. Factors limiting industrial adoption of test driven development: A systematic review. In *Software Testing, Verification and Validation (ICST), 2011 IEEE Fourth International Conference on*, pages 337–346. IEEE, 2011.

[8] A. Čaušević, D. Sundmark, and S. Punnekkat. Impact of test design technique knowledge on test driven development: A controlled experiment. In *Agile Processes in Software Engineering and Extreme Programming*, pages 138–152. Springer, 2012.

[9] J. Cohen and P. Cohen. *Applied multiple regression/correlation analysis for the behavioral sciences*. Lawrence Erlbaum, 1975.

[10] J. E. Cook and A. Wolf. Toward metrics for process validation. In *Software Process, 1994. 'Applying the Software Process', Proceedings., Third International Conference on the*, pages 33–44, 1994.

[11] W. Cunningham and R. C. Martin. Praise for growing object-oriented software, guided by tests.

[12] S. H. Edwards. Using test-driven development in the classroom: Providing students with automatic, concrete feedback on performance. In *Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA*, volume 3, 2003.

[13] H. Erdogmus, M. Morisio, and T. Marco. On the Effectiveness of the Test-First Approach to Programming. *IEEE Transactions on Software Engineering*, 31(3):226–237, 2005.

[14] 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*, 4:1, 2010.

[15] M. Fowler. *Refactoring: improving the design of existing code*. Addison-Wesley Professional, 1999.

[16] M. Fowler. The new methodology. *Zugriff*, 4:2007, 2005.

[17] M. Fowler. Test-driven development. *Available on: http://www. martinfowler. com/bliki/TestDrivenDevelopment. html*, 6, 2011.

[18] S. Fraser, D. Astels, K. Beck, B. Boehm, J. McGregor, J. Newkirk, and C. Poole. Discipline and practices of tdd:(test driven development). In *Companion of the 18th annual ACM SIGPLAN conference on Object-oriented programming, systems, languages, and applications*, pages 268–270. ACM, 2003.

[19] S. Fraser, K. Beck, B. Caputo, T. Mackinnon, J. Newkirk, and C. Poole. Test driven development (tdd). In *Extreme Programming and Agile Processes in Software Engineering*, pages 459–462. Springer, 2003.

[20] D. Fucci and B. Turhan. On the role of tests in test-driven development: a differentiated and partial replication. *Empirical Software Engineering*, pages 1–26, 2013.

[21] D. Fucci and B. Turhan. A replicated experiment on the effectiveness of test-driven development. In *to appear in Proceedings of the ACM-IEEE international symposium on Empirical software engineering and measurement*. ACM, 2013.

[22] B. George and L. Williams. A structured experiment of test-driven development. *Information and Software Technology*, 46(5):337–342, 2004.

[23] G. V. Glass. Testing homogeneity of variances. *American Educational Research Journal*, 3(3):187–190, 1966.

[24] T. Henchy and D. C. Glass. Evaluation apprehension and the social facilitation of dominant and subordinate responses. *Journal of Personality and Social Psychology*, 10(4):446, 1968.

[25] J. R. Horgan, S. London, and M. R. Lyu. Achieving software quality with testing coverage measures. *Computer*, 27(9):60–69, 1994.

[26] D. S. Janzen and H. Saiedian. Does test-driven development really improve software design quality? *Software, IEEE*, 25(2):77–84, 2008.

[27] P. Johnson and H. Kou. Automated Recognition of Test-Driven Development with Zorro. In *AGILE 2007*, pages 15–25. IEEE, 2007.

[28] C. Kaner. Software negligence and testing coverage. *Software QA Quarterly*, 2(2), 1996.

[29] M. I. Kellner, P. H. Feiler, A. Finkelstein, T. Katayama, L. J. Osterweil, M. H. Penedo, and H. D. Rombach. Software process modeling example problem. In *Software Process Workshop, 1990.'Support for the Software Process'., Proceedings of the 6th International*, pages 19–29. IEEE, 1990.

[30] H. Kou, P. M. Johnson, and H. Erdogmus. Operational definition and automated inference of test-driven development with zorro. *Automated Software Engineering*, 17(1):57–85, 2010.

[31] W. Krebs. Turning the knobs: A coaching pattern for xp through agile metrics. In *Extreme Programming and Agile MethodsÑXP and Agile Universe 2002*, pages 60–69. Springer, 2002.

[32] L. Madeyski. The impact of pair programming and

test-driven development on package dependencies in object-oriented design — an experiment. *Product-Focused Software Process Improvement*, pages 278–289, 2006.

[33] L. Madeyski and Ł. Szała. The impact of test-driven development on software development productivity — empirical study. In *Software Process Improvement*, pages 200–211. Springer, 2007.

[34] Y. Malaiya, M. Li, J. Bieman, and R. Karcich. Software reliability growth with test coverage. *Reliability, IEEE Transactions on*, 51(4):420–426, 2002.

[35] Y. K. Malaiya, N. Li, J. Bieman, R. Karcich, and B. Skibbe. The relationship between test coverage and reliability. In *Software Reliability Engineering, 1994. Proceedings., 5th International Symposium on*, pages 186–195. IEEE, 1994.

[36] B. Marick. How to misuse code coverage. In *Proceedings of the 16th Interational Conference on Testing Computer Software*, pages 16–18, 1999.

[37] R. C. Martin. Professionalism and test-driven development. *Software, IEEE*, 24(3):32–36, 2007.

[38] A. Montenegro. On sample size and precision in ordinary least squares. *Journal of Applied Statistics*, 28(5):603–605, 2001.

[39] R. Mugridge. Challenges in teaching test driven development. In *Extreme Programming and Agile Processes in Software Engineering*, pages 410–413. Springer, 2003.

[40] M. Müller and A. Höfer. The Effect of Experience on the Test-driven Development Process. *Empirical Software Engineering*, 12(6):593–615, 2007.

[41] B. d. S. C. Pedroso. Besouro: aprimorando a aferição automática da conformidade das atividades de desenvolvimento com tdd. 2012.

[42] E. A. Pena and E. H. Slate. Global validation of linear model assumptions. *Journal of the American Statistical Association*, 101(473):341–354, 2006.

[43] M. Philipp. Comparison Of The Test-Driven Development Processes Of Novice And Expert Programmer Pairs. 2009.

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

[45] W. R. Shadish, T. D. Cook, and D. T. Campbell. Experimental and quasi-experimental designs for generalized causal inference. 2002.

[46] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus. What Do We Know About Test-driven Development? *Software, IEEE*, 27(6):16–19, 2010.

[47] L. Silva and G. Travassos. Tool-supported unobtrusive evaluation of software engineering process conformance. In *Empirical Software Engineering, 2004. ISESE '04. Proceedings. 2004 International Symposium on*, pages 127–135, 2004.

[48] S. Sørumgård. Verification of process conformance in empirical studies of software development. *Department of Computer and Information Science, The Norwegian University of Science and Technology*, 1997.

[49] M. Srivastava and T. Hui. On assessing multivariate normality based on shapiro-wilk statistic. *Statistics & Probability Letters*, 5(1):15–18, 1987.

[50] Y. Wang and H. Erdogmus. The role of process measurement in test-driven development. In *4th Conference on Extreme Programming and Agile Methods*, 2004.

[51] D. Wells and L. Williams. *Extreme Programming and Agile Methods-XP/Agile Universe 2002: Second XP Universe and First Agile Universe Conference Chicago, IL, USA, August 4-7, 2002. Proceedings*, volume 2418. Springer, 2002.

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

[53] N. Zazworka, V. Basili, and F. Shull. Tool supported detection and judgment of nonconformance in process execution. In *Empirical Software Engineering and Measurement, 2009. ESEM 2009. 3rd International Symposium on*, pages 312–323, 2009.

[54] N. Zazworka, K. Stapel, E. Knauss, F. Shull, V. R. Basili, and K. Schneider. Are developers complying with the process: an xp study. In *Proceedings of the 2010 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement*, ESEM '10, pages 14:1–14:10, 2010.

[55] K. Zieliriski and T. Szmuc. Preliminary analysis of the effects of pair programming and test-driven development on the external code quality. *Software engineering: evolution and emerging technologies*, 130:113, 2006.