The Effects of Test-Driven Development on External Quality and Productivity: A Meta-Analysis

Yahya Rafique and Vojislav B. Mišić, *Senior Member*, *IEEE*

Abstract—This paper provides a systematic meta-analysis of 27 studies that investigate the impact of Test-Driven Development (TDD) on external code quality and productivity. The results indicate that, in general, TDD has a small positive effect on quality but little to no discernible effect on productivity. However, subgroup analysis has found both the quality improvement and the productivity drop to be much larger in industrial studies in comparison with academic studies. A larger drop of productivity was found in studies where the difference in test effort between the TDD and the control group's process was significant. A larger improvement in quality was also found in the academic studies when the difference in test effort is substantial; however, no conclusion could be derived regarding the industrial studies due to the lack of data. Finally, the influence of developer experience and task size as moderator variables was investigated, and a statistically significant positive correlation was found between task size and the magnitude of the improvement in quality.

Index Terms—Test-driven development, meta-analysis, code quality, programmer productivity, agile software development

1 Introduction

Test-driven Development (TDD) is among the cornerstone practices of the Extreme Programming (XP) development process [1], [2], [3] and today is being widely adopted in industry both as part of a large-scale adoption of XP and as a stand-alone practice. TDD is commonly considered to be the amalgamation of test-first development, in which unit tests are written before the implementation code needed to pass those tests [2], and refactoring, which includes restructuring a piece of code that passes the tests in order to reduce its complexity and improve its clarity, understandability, extendibility, and/or maintainability [3]. TDD is often described with the so-called "red-green-refactor cycle" [4] that consists of the following steps:

1. Design and add a test.
2. Run all tests and see the new one fail (*red*).
3. Add enough implementation code to satisfy the new test.
4. Run all tests, repeat 3 if necessary until all tests pass (*green*).
5. Occasionally *refactor* to improve code structure.
6. Run all tests after refactoring to ensure all tests pass.

The use of TDD is claimed to bring improvements in code quality and productivity. However, research studies investigating the effectiveness of TDD have failed to produce conclusive results; in fact, all possible outcomes—positive, negative, and neutral—have been reported for both quality and productivity improvements obtained with TDD.

In this paper, we present a systematic review of 27 empirical studies that have been published until February 2011. Our review is limited to aggregating empirical findings on two, arguably most important, outcome constructs: external code quality (hereafter referred to as quality), expressed through the number of defects per given code size unit (such as lines of code or another suitable measure), and productivity, expressed as the number of given units produced in a given time frame (day or month) [5]. Studies which focus on other outcome constructs such as internal quality or system design, test coverage, etc., have been excluded from the forthcoming analysis, as will be explained below.

For each study, we begin by analyzing the experimental details. Then, based on these details, we discuss its perceived rigor and the impact that differences in rigor might have had on the studies' results related to quality and productivity. After that, we compute individual effect-sizes for each analyzed experiment and combine them into a summary value using meta-analytical techniques [6]. Finally, we investigate the impact of developer experience and task size as moderator variables.

While every effort was made to include all relevant studies, this review is limited to studies that compare the performance of TDD with that of a more conventional development process in an empirical setting. In this process, we have tried to adhere as much as possible to the general guidelines for reviews in software engineering [7]. Furthermore, as a step toward unifying research in the stream of agile development, the layout of this paper is similar to that of a recent meta-analysis on Pair Programming [8].

The paper is organized as follows: Section 2 provides a description of the methodology used to gather studies,

- *Y. Rafique is with Ryerson University, 96 Saint Patrick Street, Suite #TH101, Toronto, ON M5T 1V2, Canada. E-mail: yahya.rafique@ryerson.ca.*
- *V.B. Mišić is with Ryerson University, 350 Victoria Street, Toronto, ON M5B 2K3, Canada. E-mail: vmisic@ryerson.ca.*

Manuscript received 12 May 2011; revised 7 Mar. 2011; accepted 8 Apr. 2012; published online 2 May 2012.
Recommended for acceptance by L. Williams.
For information on obtaining reprints of this article, please send e-mail to: tse@computer.org, and reference IEEECS Log Number TSE-2011-05-0146.
Digital Object Identifier no. 10.1109/TSE.2012.28.

TABLE 1
Studies Not Included in This Review

Reason for exclusion	Study
Lack of quantitative data.	[10] [11] [12]
No control group.	[13] [14] [15] [16]
Different outcome constructs considered.	[17] [18] [19] [20] [21] [22] [23]
Lack of process conformance.	[24]: tests are written 'before or in conjunction with' source code
	[25]: focus on automating 'component-level' (i.e., higher level) tests
	[26]: both the TDD and the traditional approach are applied to the same source code
Repeats the results of another study.	[27]: same as one of the experiments in [28] (included)
	[29]: same experiment as [30] (included)
	[31]: same experiment as [32] (included)
	[33]: same as one of the experiments in [28] (included)
	[34]: same experiment as [35] (included)
	[36]: same experiment as [15] (excluded)
	[37]: same as two of the experiments in [38] (included)
	[39]: same experiment as [40] (included)
	[41]: same as one of the experiments in [42] (included)

extract the results, and compute respective effect sizes, and synthesize these effect sizes into a summary effect size for each construct. Section 3 presents a description of the selected experiments including an analysis on rigor. Results related to the external quality construct are given in Section 4 and results related to the productivity construct are given in Section 5. Section 6 analyzes the impact of two potential moderator variables, namely, developer experience and task size. Threats to the validity of the analysis are given in Section 7. Section 8 provides a comparison between this review and three earlier reviews. Finally, Section 9 concludes by suggesting some promising directions for future research.

2 Methodology

The meta-analysis procedure has gained considerable attention in recent years as one of the effective ways to quantitatively summarize and, if possible, interpret the results of a collection of single studies on a given topic [9]. The analysis proceeds through a number of distinct steps, as follows.

2.1 Study Identification and Selection

The identification and selection process proceeded in three stages. First, we identified candidate studies by querying the electronic databases of the ACM Digital Library, IEEE Xplore, SpringerLink, ISI Web of Science, and Scopus, using the strings "Test Driven Development," "Test First Development," and "TDD" to search through the Article Title, Abstract, and Keyword fields. The generated matches were filtered to include only studies published in peer-reviewed journals or proceedings from peer-reviewed conferences. The resulting matches were prescreened for relevance by reading through the titles and abstracts but also, in some cases, going through the introduction. All studies found to be relevant, as well as those whose relevance was still unclear, were selected for a more thorough analysis.

In the final stage, each of the authors read all of the selected studies and individually compiled a list of studies to be included in the review. The individual lists were then compared and all differences were resolved through discussion. Accordingly, a final list of studies was derived which would form the subject of the upcoming meta-analysis.

2.2 Inclusion and Exclusion Criteria

Studies were included in this meta-analysis if they reported results on one or more experiments in which the effectiveness of TDD was compared with that of a more traditional (i.e., Test-Last) approach. Such experiments were designed with subjects being divided into two or more groups, each of which developed the same or similar products with at least one group following either development approach. Studies were only included if they reported quantitative data on at least one of the investigated outcome constructs. The use of other agile practices along with TDD was not considered as a limiting factor; however, it is recognized as a threat to validity, as explained in Section 7. We also note that the choice of metrics used to measure the outcome constructs was not a mandatory requirement for inclusion, as it was desirable to incorporate as many relevant studies as possible.

During the selection process, a number of studies were deemed unsuitable for inclusion in our review due to one or more of the following reasons:

- Some studies provided data obtained only with TDD, possibly over multiple releases of the same product, but without a control group.
- Some studies lacked a quantitative component, relying instead on qualitative assessment or practitioner perceptions.
- Other studies focused on other outcome constructs such as design quality or test coverage. However, such measures are, at best, only indirect measures of quality and there is no unified metric for either of them. (In particular, design quality is notoriously difficult to assess.) Therefore, we decided against using design quality as an outcome construct.
- In some cases, the development process did not follow TDD with sufficient rigor: by applying TDD and traditional approach to the same code, or by allowing tests in the treatment group to be authored before or in conjunction with code.
- Finally, some studies simply repeated the results of others, either partially or in their entirety.

The list of studies that were excluded at the final stage of the selection process, along with the reasons for their exclusion, is given in Table 1. We hasten to add that the

exclusion does not imply that these studies are without merit; it simply means that the approach taken in these studies did not align well with the goals of this analysis.

2.3 Data Extraction and Output Categories

The data extracted from the studies was classified into three categories: Context, Rigor, and Outputs.

Attributes in the first category recorded contextual and other high-level details regarding the studies, including the authors of the study, the number of participants, and the context—academia or industry—in which the experiment was conducted.

Attributes in the Rigor category aimed to help assess the extent of the applicability of a study's results according to the criteria for study rigor described in [43]. These attributes include the following:

- *CT*, which indicates the manner in which testing was done by the control group—iteratively, i.e., interleaved with coding, or after the target system was fully implemented—as discussed in Section 3.3;
- *OA*, indicating the other agile practices that were included in the development processes;
- development and programming experience of the subjects;
- task size of the final application (in LOC);
- duration of the project;
- information about process conformance in the target group (i.e., adherence to the widely accepted principles of TDD development);
- details of training received by the subjects prior to the experiment.

The last category, Outputs, contained attributes for the format in which results of a study were reported. For some experiments, the authors reported the mean and standard deviation of the results of each of the subject groups, and perhaps a p-value, which allows tests of statistical significance to be run. For others, the results included only the percentage improvement for each group or (in the worst case) just a single value for an outcome construct, which was then simply transformed into a percentage improvement value.

Our original intention was to compute a standardized effect size for each of the analyzed experiments and then synthesize individual experiment effect sizes into a summary effect size using one of the meta-analytical statistical models, as explained below. However, this requires that the mean and standard deviation for each subject group are available—which, unfortunately, was not the case with all studies. As our goal was to include as many studies as possible, we decided to split the analysis into two subanalyses, dubbed Standardized and Unstandardized. For the former, standardized effect sizes were calculated when possible and subsequently combined using the meta-analytical models explained below. For the latter, an unstandardized effect size was calculated for each analyzed experiment; the individual effect sizes were simply averaged to obtain the summary effect size. Therefore, standardized and unstandardized analyses differ in their choice of effect size measure and the manner in which individual effect sizes are synthesized into a summary value, both of which are explained in detail below.

2.4 Standardized Analysis

All standardized effect sizes in this paper were computed using the Comprehensive Meta-Analysis V2 tool by BioStat, Inc [44]. The Hedges' g statistic was chosen as the standardized effect size measure for the analysis as it exhibits better characteristics for smaller samples when adjusted for small sample bias in comparison with other parametric measures such as Cohen's d and Glass' Δ [8]. The Hedges' g statistic is calculated as

$$g = \frac{m_t - m_c}{s_{pooled}}, \tag{1}$$

where m_t and m_c refer to the mean values reported for the treatment and control groups, respectively, and s_{pooled} refers to the pooled standard deviation, which is calculated from the standard deviation in the treatment group s_t and the control group s_c as

$$s_{pooled} = \sqrt{\frac{(n_t - 1)s_t^2 + (n_c - 1)s_c^2}{(n_t - 1) + (n_c - 1)}}. \tag{2}$$

Variables n_t and n_c denote the number of subjects in the treatment and control groups, respectively. The variable n_{total} is the sum of n_t and n_c and is given in the "subjects" column in Tables 2 and 3.

Small sample bias is accounted for by multiplying the g statistic by a correction factor of

$$cf = 1 - \frac{3}{4(n_{total} - 2) - 1}. \tag{3}$$

According to [45], effect sizes with a value in the range 0.0-0.37, 0.38-1.0, and 1.0 and above can be considered as small, medium, and large sized effects, respectively. Positive effect sizes represent an improvement as a result of applying the treatment, whereas negative values imply a detrimental impact.

Individual experiment effect sizes were combined to obtain a summary effect size using two popular statistical models, namely, the fixed-effects model and the random-effects model [46]. The principal difference between these two models is the underlying assumption. The fixed-effects model assumes that only one true effect size underlies all individual experiment effect sizes and differences among them are solely due to sampling error, i.e., as the sample size increases, the effect size is likely to converge to the one true value. On the other hand, the random-effects model is premised on the assumption that differences among individual effect sizes are the result of the sampling error as well as of other variables and factors that have not yet been taken into account. Consequently, the effect size may vary from study to study, and is distributed about some mean effect size value. The two models thus attempt to answer slightly different questions: The fixed-effects model aims to derive the one true effect size, while the random-effects models aim to find the mean of the curve along which all possible effect sizes are distributed.

However, both models follow a similar procedure to derive the summary effect size. First, for each experiment i (in the k experiments) they assign a weight w_i. Then, the summary effect size T_s is computed as the weighted average of the individual experiment effect sizes T_i:

TABLE 2
Details of Academic Studies

experiment	subjects	CT[a]	OA[b]	experience	size (LOC)	duration	conformance	training	other details
Desai *et al.* [50]	166[c]	W		junior UG		10 hrs[d]		Lab materials provided instructions on TDD, prepared by a person with limited TDD experience and no curriculum development experience.	Experiment comprised seven projects dealing with writing shape classes like Triangle, Rectangle etc. Introductory Java course.
Edwards [32]	118	W		senior UG		12 weeks[e]		Students were given 30 minutes of classroom training on TDD and testing.	Task: four assignments. Comparative Languages course.
Erdogmus *et al.* [5]	24	I		intermediate UG	272[f]	32 hrs[g]	Post-survey and tests used to judge conformance for each subject.	Students trained in Test-First and Test-Last techniques.	Task: bowling score keeper. Introductory Java course.
Flohr and Schneider [51]	18	I	PP, OSC	G		23 hrs[h]	Process conformance was poor.	A 2 hour lecture on TDD and testing. More training was needed.	Task: library for a system aimed to provide graphical description of communications flows in software processes.
George E1 [28]	138	W	PP	junior, senior UG	200	1.25 hrs		Students were taught TDD and JUnit. Two homework assignments completed with TDD and JUnit before experiment.	Task: bowling score keeper. Software Engineering class.
Gupta & Jalote E1 [47]	22	W		G, senior UG	1600	31.5 hrs	In the post-survey, 70% of TDD group and 88% of Traditional group said they adhered to their approach.	Subjects given necessary training to use TDD for developing Java programs; however, in the post-survey 47% of TDD group stated they need more training.	Student course registration system. Advanced OO analysis and modeling course.
Gupta & Jalote [47] E2	22	W		G, senior UG	1600	34.5 hrs	see above	see above	Simple ATM system, same course as above.
Huang & Holcombe [52]	39	I	XP	intermediate G	1175	110 hrs	Project Manager was appointed to ensure conformance.	10 hours of advanced training of TDD and other XP practices.	Different projects resembling realistic tasks. Software engineering course.
Janzen E1, E5 [42]	E1: 10, E5: 9	I		E1: intermediate & senior UG, E5: G	E1: 650 E5: 760[l]	E1: 40 hrs E2: 97 hrs		Lecture slides on automated testing and TDD and sample programs. 1.5 - 2 hr training session.	Task: HTML pretty print system. E1: Undergraduate Software Engineering Course. E5: Graduate Software Engineering Course.
Janzen E2 [42]	28[j]	I		junior UG		6 weeks[k]		see above	Task: Data Structures for modeling points and polygons. CS1 Programming Course.
Janzen E3, E4 [42]	E3: 36, E4: 54	I		junior UG		9 weeks[l]		see above	Task in E3: Application for tracking drivers and traffic citation information. Task in E4: Similar to E3 but using airline flight information. Both E3, E4 done in CS2 Programming Course.
Kaufmann & Janzen [53]	8	W[m]		UG			Both groups did not write sufficient amount of tests	Training given on Graphics APIs utilization	Task: Small games in Java using Graphics APIs. Elective Software Studio course.
Madeyski E1 [48]	56	I		intermediate, senior UG		12 hrs		The course introduced Java using TDD and pair programming as the key XP practices. Sample programs were also included.	Finance accounting system. Introductory Java course.
Madeyski E2 [48]	132	I	PP	see above		see above		see above	see above
Muller & Hagner [54]	19	W		G				The course in which experiment was conducted covered pair programming, TDD, refactoring and planning techniques.	Main class of a graph library. Second course on XP.

$$T_s = \sum_{i=1}^{k} w_i T_i \Big/ \sum_{i=1}^{k} w_i. \tag{4}$$

The weights w_i are calculated as the inverse of the experiment's error variance. Because of different model assumptions, this error variance is calculated in different ways for fixed- and random-effects models [46]:

$$\begin{aligned} w_{if} &= \frac{1}{v_i}, \\ w_{ir} &= \frac{1}{v_i + \tau^2}. \end{aligned} \tag{5}$$

In the fixed-effects model, the only source of variance is the sampling process; hence the weights w_{if} are computed as the inverse of just the within-experiment variance v_i. In the

TABLE 2
Details of Academic Studies (continued from previous page)

experiment	subjects	CT[a]	OA[b]	experience	size (LOC)	duration	conformance	training	other details
Pančur *et al.* [55]	38	I	PP	senior UG					2 assignments taken over a period of less than three weeks.
Pančur & Ciglarič E1, E3 [49]	E1: 19, E3: 16	I	PP	senior UG	E1: 10 user stories (us), E3: 13 us	26 hrs	Conformance monitored using Process Log Eclipse Plug-in.	Lectures contained 'several' practical examples on unit testing, JUnit, refactoring, agile design, frequent mistakes etc. Lectures followed by three short assignments which were assessed to provide subjects with individualized comments and warnings.	E1: Part of Series 1 E3: Part of Series 2
Pančur & Ciglarič E2, E4 [49]	E2: 23, E4: 32	I		senior UG	2 user stories	4 hrs		see above	E2: Part of Series 1 E4: Part of Series 2
Rahman [56]	150	W		junior & intermediate UG			Treatment group did detailed design.	20-25 minute presentation on experimental goals and Test-before-Coding (TBC) method. Afterwards also did a similar problem together.	Experiment conducted over two years in five introductory programming courses.
Vu *et al.* [57]	14	W		G, senior UG	3070[i]	147.7 hrs[l]		Test-First and Test-Last methodologies introduced through lectures and student presentations.	Software engineering course.
Xu & Li [58]	8	W		intermediate, senior UG	344	4 hrs	One of the authors acted as a mentor who monitored the programming process for all the individuals.	Short training sessions in which students were given reading materials and asked to implement a simple program to get used to of procedure and tools.	Task: bowling score keeper. Software engineering course.
Yenduri & Perkins [59]	18	W		senior UG				Students were given required user manuals and a short description of forming test cases. They were also trained in applying both TDD and traditional approaches.	Software engineering course.
Zhang *et al.* [60]	8	W		G, UG[n]		June to Aug. 2005			Working attendance management system.

Acronyms - E#: Experiment #, UG: Undergraduate, G: Graduate, PP: Pair Programming, OSC: On-site Customer, XP: Extreme Programming(represents all practices within the XP methodology)
[a] CT: testing approach in the control group. 'W' denotes Waterfall (no test written before all the code has been finished; 'I' denote ITL (Iterative Test-Last), testing interleaved with coding but with test written after the code to be tested.
[b] OA: presence ('Y' or 'N') of other agile practices, such as pair programming.
[c] Assuming the 2007 class and 2008 class had the same number of subjects.
[d] Average duration per project determined by combining hours spent by both control and treatment groups.
[e] Four assignments each of which takes 2 to 3 weeks to complete.
[f] Estimated using average of George & Williams study and Xu & Li's study since all three worked on implementing same problem.
[g] 4 hrs per week in a 8 week course.
[h] Average based on individual subject attempts.
[i] Average of treatment and control groups.
[j] Number of students who took post-experiment survey.
[k] Two assignments with each lasting 2 to 3 weeks.
[l] Each experiment consisted of three assignments with each assignment lasting 2 to 3 weeks.
[m] Experiment included in ITL subgroup in forthcoming analysis because test effort was similar to TDD group.
[n] Subjects referred to simply as students without highlighting seniority.

random-effects model, other factors could also add on to the overall variance. Consequently, the weights w_{ir} are computed as the inverse of the sum of the within-experiment variance v_i and a constant τ^2 representing the between-experiment variance, which is calculated using the DerSimonian and Laird formula [8]:

$$\tau^2 = \begin{cases} \dfrac{Q - df}{C}, & \text{if } Q > df, \\ 0, & \text{if } Q \leq df. \end{cases} \tag{6}$$

The degrees of freedom df is equal to the number of individual experiment effect sizes minus one. The Q statistic is the weighted sum of the squares of the deviations of each experiment's effect size from the summary effect size:

$$\begin{aligned} df &= k - 1, \\ Q &= \sum_{i=1}^{k} w_{if} \left(T_i - T_{sf}\right)^2. \end{aligned} \tag{7}$$

TABLE 3
Details of Industrial Studies

experiment	subjects	CT[a]	OA[b]	experience (years)	size (LOC)	duration	conformance	training	other details
Canfora *et al.* [40]	28	W		5		10 hrs		3 hr training session included a seminar on TDD and lab exercises to increase familiarity with the practice.	TextAnalyzer system.
Dogša & Batič [61]	36	I	PP	5	57,169[c]	20,834 hrs[c]	For both groups the test suite automatically ran every 15 minutes and sent emails on failed tests. With TF group no evidence of even any 'little-design-up-front.'	'Intensive' training on TF approach by solving 'carefully selected real-world' problems for three weeks.	Task: A simulator for a node in a IP multimedia subsystem architecture designed by the wireless standards body 3GPP.
George E2 [28]	24	W	PP		200	5 hrs[d]			Task: Bowling Score Keeper application. Groups in three companies.
Geras *et al.* [35]	14	I		0.5 to 6	'small'		Subjects followed previously prepared process scripts.		Task: Project Time Entry System.
Lui & Chan [62]		W		-[e]					Custom software for manufacturing plants in less developed areas in China.
Nagappan *et al.* E1 [38]	8	W		6 to 10[f]	6000	3840 hrs[g]	Treatment group did detailed design.		Networking common library.
Nagappan *et al.* E2 [38]	20	W		6 to 10[f]	26000	7360 hrs[g]	see above		Web Service application.
Nagappan *et al.* E3 [38][h]	12	W		10+[f]	155200	3200 hrs[g]	see above		Part of the development of an IDE tool.
Slyngstad *et al.* [63]	100	W			14671[i]				Five releases of an internally reusable framework. Developers distributed worldwide, mostly in Norway and Sweden.
Williams *et al.* [30]	14	W			64000			Dedicated TDD coach was assigned as technical leader.	Subjects developed a release of a device driver.

Acronyms - E#: Experiment #, PP: Pair Programming
[a] CT: testing approach in the control group. 'W' denotes Waterfall (no test written before all the code has been finished; 'I' denote ITL (Iterative Test-Last), testing interleaved with coding but with test written after the code to be tested.
[b] OA: presence ('Y' or 'N') of other agile practices, such as pair programming.
[c] Average of Test-Last and both Test-First teams.
[d] Approximated from box plots and average of TDD and control groups.
[e] Study states that developers were inexperienced.
[f] Majority of the developers had this experience level.
[g] Assuming they worked 160 hrs per month.
[h] Only three of the four experiments mentioned in the paper are included; the fourth one is already included in [30].
[i] Average size of a release.

In order to obtain Q, the weights w_{if} and the summary effect size T_{sf} must be computed beforehand, following the fixed-effects model. Finally, C is a scaling factor used to ensure that τ^2 is reported in the same metric as the within-experiment variance v_i:

$$C = \sum_{i=1}^{k} w_i - \sum_{i=1}^{k} w_i^2 \Big/ \sum_{i=1}^{k} w_i. \quad (8)$$

The precision V_{sf} with which the summary effect size T_{sf} in the fixed-effects model estimates the one true effect size is

$$V_{sf} = 1 \Big/ \sum_{i=1}^{k} w_i. \quad (9)$$

The choice of the model is made on the basis of factors such as knowledge of the environment, previous empirical observations and findings, and the like, and some assistance is provided by the modeling process itself. In particular, the Q statistic from (7) is consistent with the level of heterogeneity [8]; hence a significant value of Q should support the rejection of the homogeneity hypothesis and the adoption of a random-effects model. We have also computed the I^2 statistic:

$$I^2 = \frac{Q - df}{Q} \times 100\%, \quad (10)$$

which is used to quantify the degree of true heterogeneity, i.e., the extent to which the total variance is the result of between-experiment variance [9]. I^2 values of 33.3 and 66.6 percent delimit low, moderate, and high level of heterogeneity.

2.5 Unstandardized Analysis

Percentage improvement was chosen as the unstandardized effect size measure, according to the formula

$$\Delta x = \frac{x_t - x_c}{x_c} \times 100\%. \quad (11)$$

The exact values for x_t and x_c depend on the design of the experiment. All of the analyzed studies had adopted one of two designs for their experiments, which will be explained next. The difference in design is the primary reason why results were reported differently, as explained above in the description of the Outputs category of attributes.

In the first design, each group of subjects jointly developed a system. For each investigated outcome construct, the authors had reported a single value for each subject group and/or went a step further and computed the percentage improvement. If only the percentage improvement was reported, then it was accepted without change, but if the authors had reported actual group values, the percentage improvement was recalculated using (11), with the variables x_t and x_c set to the values reported for the treatment and control groups, respectively.

In the second design, all subjects in a group individually developed the same system. In this case, for each investigated outcome construct the study authors had either reported individual values for each subject in each group and/or they had reported the mean and the standard deviation for each group. In this design the means, or medians when the means were not given, for the subject groups were used to set the variables x_t and x_c.

Consequently, only a single observable data point or effect size was recorded for each experiment regardless of the experimental design; however, the method to calculate the effect size, as indicated above, was dependent on the experimental design. Using a single effect size per experiment was the case not only in the unstandardized analysis but also in the standardized analysis. Exceptions where results of two experiments were combined to give a single experiment effect size are explained below.

The summary effect size was then calculated as the mean of the individual effect sizes.

3 Description of Experiments

3.1 Listing of Experiments

For this review a total of 37 experiments were extracted from 25 studies which were selected based on the criteria in Section 2.2. Typically, studies had reported on only a single experiment; in this case the effect size realized in the experiment was calculated as explained above. However, some studies reported results on multiple experiments and these were treated differently, as is explained next.

For studies that reported results on up to three distinct experiments, an individual effect size was calculated for each experiment and, consequently, the study contributed more than one effect size value toward the forthcoming synthesis process. This was the case with the studies by Gupta and Jalote [47], Madeyski [48], and Nagappan et al. [38], which contributed two, two, and three effect size values, respectively.

For studies that reported results on more than three experiments, namely, those by Janzen [42] and Pančur and Ciglarič [49], an effect size was not calculated for an individual experiment but for the combined or average result of a group of experiments instead. The limit of three experiments was intended to restrict the number of effect sizes contributed by a study in order to avoid publication bias. However, for studies reporting results of up to three experiments, individual experiment effect sizes were calculated with the underlying intent of maximizing the number of effect sizes included in the synthesis process and thus benefiting from a more accurate summary effect size value. In other words, the limit of three experiments is simply an attempt to strike a balance between maximizing the number of effect sizes and avoiding a publication bias toward any study.

A personalized grouping strategy was chosen for each of the two studies in which the experiments were to be grouped. In the study by Janzen [42], E1 (i.e., Experiment #1) and E5 were grouped as the same development task was undertaken in each both of these experiments. Also, in the same study E3 and E4 were grouped as they had a very similar development task according to the author. Having reported the results of five experiments, after the grouping process the study by Janzen [42] contributed three effect size values to the forthcoming synthesis process. In the study by Pančur and Ciglarič [49], E1 and E3 were grouped and so were E2 and E4 as the task size among the experiments in each pair was the same or relatively similar, but the difference across pairs was significant. Consequently, this study contributed two effect size values to the synthesis process.

Details of the academic and industrial experiments analyzed in this review are given in Tables 2 and 3, respectively. The experiments are labeled according to the authors of the corresponding report, with the suffix "Ei," where $i = 1, 2, \ldots,$ added where the report contained data on multiple experiments. Experiments reported in the two studies [42], [49], which were grouped together to meet the publication bias constraint, are described as a single entry per group.

3.2 Metrics Used

The choice of metrics used to operationalize the outcome constructs differed among the experiments. According to [8], the use of different metrics results in the experiments analyzing slightly different aspects of the outcome constructs. As such, any metric conveys but one out of many possible viewpoints regarding the impact on these outcome constructs. Therefore, the choice of a specific metric might be among the reasons that lead to differences in individual experiment effect sizes. Unfortunately, the lack of a widely accepted standard in this area is likely to exist for the foreseeable future.

The quality construct was operationalized using one or more of the following metrics:

- number of defects/bugs/trouble reports/defect cases that were found,
- defects per KLOC/defect density,
- number/percentage of acceptance/black-box/external tests passed,
- number/percentage of unit tests passed, and
- quality mark given by client.

The productivity construct was operationalized using one or more of the following metrics:

- development time/person hours spent/task time,
- total LOC divided by total effort, or the number of LOC per hour,
- total noncommented LOC,
- number of delivered stories per unit effort (or implemented user stories per hour),
- delivered noncommented LOC per unit development effort (or effort per ideal programming hour), and
- hours per feature/development effort per LOC.

3.3 Assessment of Study Rigor

Studies differed in their definitions of the control group's process. As noted in the "CT" column in Tables 2 and 3, for the control group studies were found to either opt for a Waterfall-based testing process (labeled "W"), in which testing is done after all functionality has been implemented, in effect mimicking the Waterfall development process, or the Iterative Test-Last (ITL) development approach [49] ("I"), in which unit testing is interleaved with coding as in TDD, but the unit tests are written after the implementation code. As a direct implication of the testing process chosen, studies were found to differ in the amount of test effort expended by the control group. In particular, control group subjects adopting the ITL approach were generally found to spend more time testing than their counterparts employing the Waterfall approach. This can primarily be seen as the result of the ITL constraint of writing tests after every little snippet of functionality, which in effect obligated developers to spend some time testing after short cycles of development, thus avoiding pitfalls of Waterfall-based development where testing, in the end, might be ignored due to reasons such as tiredness, close deadlines, etc.

Studies also differed in their definitions of the treatment group's process, i.e., TDD, and other pertinent aspects such as the granularity of the tests, importance given to refactoring, frequency with which tests were run, and other minor details.

Studies also differed in the use (or lack) of other agile practices such as pair programming; this information is indicated in the "OA" column in Tables 2 and 3. The presence of other agile practices might confound the effects of TDD, which is why our initial screening process aimed at identifying studies that explicitly focused on examining the effectiveness of TDD alone, as explained in Section 2.2. Still, our meta-analysis did include some of the studies that used other agile practices, usually as part of both the treatment and the control groups' process definitions so that their impact on results can be minimized.

Considerable variation in the experience level of the subjects was observed. In academic studies, subjects ranged from junior undergraduate to graduate students; in industrial studies, reported (industrial programming) experience ranged from 1 to over 10 years. Intuitively, to be able to accurately compare TDD with a more conventional process, subjects should possess the technical skill set required to proficiently employ either process. It is worth noting that TDD requires a broader set of skills: in testing and refactoring as well as in programming, but unfortunately, only a few of the studies reported relevant data. Hence the degree to which the subjects met the skill requirements, in particular those of TDD, is uncertain. The effect of experience when applying TDD is studied in more detail in Section 6.

In most of the studies, subjects had little or no prior exposure to the TDD practice, and as such, the proficiency with which they used TDD would have been highly dependent on the training they received prior to the experiment. The reported extent of the training ranged from the distribution of short supplementary handouts to dedicated lecture sessions on TDD; in some cases, a dedicated coach was made available to mentor and/or overlook the entire development process. Nevertheless, it is plausible that some of the subjects did not fully grasp the essence of TDD due to the inherent difficulty of the technique. Indeed, multiple studies that coupled experiments with a practitioners' postperception survey reported that a significant percentage of the subjects had faced difficulty in applying the technique. Previous empirical research on TDD has reported that it could take developers up to two weeks to fully adapt to the TDD mindset [18]. This is a point of concern, especially in experiments of short duration.

Studies also differed in the size of the development task and/or duration of the experiment. Among the data available, task size was found to range from 200 to 155,200 LOC, and duration was found to range from 1.25 to 20,834 hours. Roughly a third of the combined entries of Tables 2 and 3 had a task size greater than 2,000 LOC, while just over half of the entries had a duration of 10 hours or more. Experiments where the development task is relatively small in size and/or short in duration are less favorable when assessing the usefulness of TDD since benefits might not be visible [64]. The relationship between the size of the development task and the magnitude of the improvement is further investigated in Section 6.

Finally, although subjects in a group were instructed to utilize a particular development process, few studies provide explicit details on actual conformance levels or on any steps that were taken to ensure a higher degree of conformance. This is important since the validity of the experiment is impaired if the developers even temporarily fall back to a traditional approach, which might happen due to time constraints, ease of technique, and other factors.

4 Results on External Quality

4.1 Aggregate Analyses

4.1.1 Standardized Analysis

Eleven standardized effect sizes were calculated for 12 experiments in which a total of 743 subjects took part. The number of experiments differs from the number of effect sizes because the results of two experiments in [49] were combined into a single experiment effect size measure, as explained in Section 3.1.

A summary of the standardized analysis on quality is shown in Table 4. The summary effect size is 0.106 and -0.010[1] under the fixed-effects and random-effects models, respectively. In both cases, the summary effect size indicates a very small to negligible impact on quality. Tests of significance confirm this: Assuming a 0.05 significance level,

1. For standardized analyses, the effect size is reported as an absolute value so as to distinguish it from the values obtained in the corresponding unstandardized analyses.

TABLE 4
Quality: Summary of Standardized Analysis

level	type	# of Effect Sizes	Model	Hedges g	95% CI	p	Q	I^2	df
0	OVERALL	11	fixed-effects	0.106	-0.056 . . 0.267	0.201	62.621	84.031	10
			random-effects	-0.010	-0.454 . . 0.433	0.964	9.768	0.000	10
1	Academic	10	fixed-effects	0.064	-0.114 . . 0.241	0.482	61.366	85.334	9
			random-effects	-0.049	-0.561 . . 0.464	0.852	8.057	0.000	9
	Industrial	1	fixed-effects	0.309	-0.082 . . 0.701	0.121	0.000	0.000	0
			random-effects	0.309	-0.082 . . 0.701	0.029	0.000	0.000	0
	Waterfall	6	fixed-effects	0.529	0.310 . . 0.748	0.000	29.640	83.131	5
			random-effects	0.301	-0.328 . . 0.930	0.348	7.030	28.876	5
	ITL	5	fixed-effects	-0.403	-0.643 . . -0.163	0.001	1.313	0.000	4
			random-effects	-0.403	-0.643 . . -0.163	0.001	1.313	0.000	4
	Agile-Inclusive	2	fixed-effects	-0.385	-0.685 . . -0.084	0.012	0.373	0.000	1
			random-effects	-0.385	-0.685 . . -0.084	0.012	0.373	0.000	1
	Agile-Exclusive	9	fixed-effects	0.305	0.114 . . 0.497	0.002	47.855	83.283	8
			random-effects	0.073	-0.451 . . 0.598	0.785	8.323	3.885	8
2	Academic+Waterfall	5	fixed-effects	0.629	0.365 . . 0.893	0.000	27.878	85.652	4
			random-effects	0.285	-0.617 . . 1.188	0.536	4.337	7.774	4
	Academic+ITL	5	fixed-effects	-0.403	-0.643 . . -0.163	0.001	1.313	0.000	4
			random-effects	-0.403	-0.643 . . -0.163	0.001	1.313	0.000	4
	Industrial+Waterfall	1	fixed-effects	0.309	-0.082 . . 0.701	0.121	0.000	0.000	0
			random-effects	0.309	-0.082 . . 0.701	0.029	0.000	0.000	0
	Industrial+ITL	0	n/a						

the p-values for both models indicate that there is no significant difference in quality between the two approaches. Heterogeneity, as shown by I^2, is high in the fixed-effects model. In view of the many differences among the studies as highlighted in Section 3.3 and the high heterogeneity observed under the fixed-effects model, the random-effects model seems to be better suited for the analysis. Consequently, for the remainder of this section and the forthcoming discussion, the quality construct is discussed in light of the random-effects model, while the data obtained with the fixed-effects model is solely provided for completeness purposes.

To facilitate comparison among individual experiment effect sizes, as well as between the individual effect sizes and the summary value, standardized effect-size analysis is often complemented with a forest plot and a one-study removed plot. In the former, the size of the square reflects the weight assigned to the experiment(s) corresponding to the square, under the random-effects model, and the length of the line passing through the square indicates the confidence interval surrounding the effect size for the experiment. In the latter, the summary effect size is shown with each study excluded; it thus serves as a measure for assessing the sensitivity of the analysis to each study. The forest plot and one-study removed plot under the random-effects model are shown in Figs. 1 and 2, respectively; for completeness, weight assignment under the fixed-effects model is shown in the rightmost row of Fig. 1.

The data in Fig. 1 shows that, according to the classification scheme from [45], only four of the 11 individual experiment effect sizes are positive, with three out of these four having a medium or higher magnitude; the other seven effect sizes are negative, and five of these are of a medium or higher magnitude. Despite more than half of the effect sizes being of medium or higher magnitude in either direction, p-values for half of the experiments indicate that there is no significant difference between the two approaches. As a side note, it can also be seen that, as a result of adopting the random-effects model, the weights are more evenly distributed in comparison with the fixed-effects model where three experiments would have accounted for more than half the weight of the analysis.

In the one-study removed plot in Fig. 2 the values next to each study under the column "Point" show the Hedges g summary effect size if the respective study is excluded from the synthesis process. Comparing the values in this column with the overall value in the last row, it can be seen that the

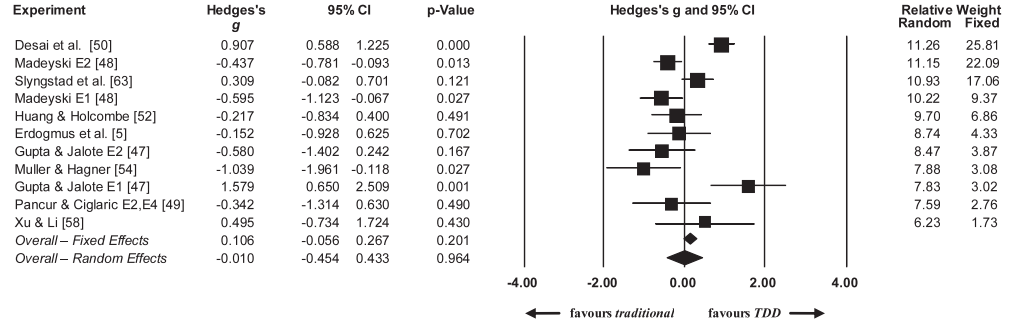


Fig. 1. Forest plot on quality under the random-effects model.

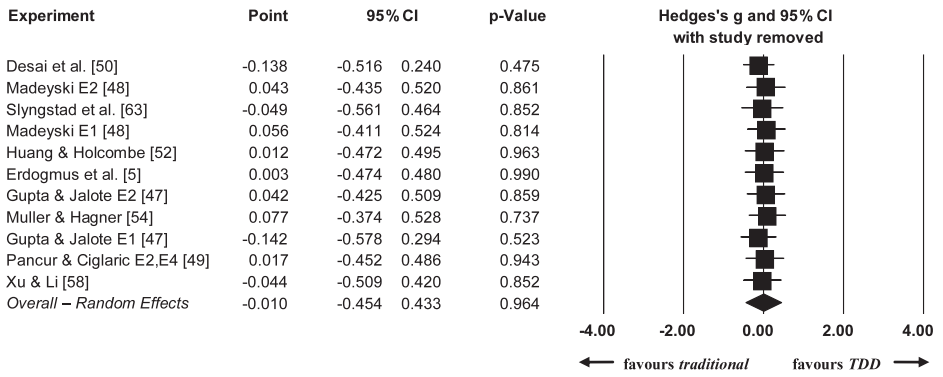


Fig. 2. One-study removed plot on quality under the random-effects model.

largest difference exists in the case of two experiments, namely, those by Desai et al. [50] and Gupta and Jalote E1 [47]. Therefore, the analysis is most sensitive to, or is most effected by, the inclusion of these two experiments. Also, with the exclusion of the sole industrial study by Slyngstad et al. [63], the one-study removed plot shows that the overall effect size of the 10 academic studies is approximately zero. A more detailed comparison between the effect sizes from academic and industrial studies is presented later on in the subgroup analyses.

4.1.2 Unstandardized Analysis

For the unstandardized analysis, 24 individual effect sizes were computed for 25 experiments which employed a total of 1,222 subjects, not counting [62], which did not report the number of subjects.

A summary of the unstandardized analysis on quality is given in Table 5 and the individual effect sizes are given in Table 6. The mean value of the improvement is 24 percent.[2] In comparison with the standardized analysis, the summary value for the unstandardized analysis illustrates a more prominent impact on quality which is largely the result of including more industrial studies into the analysis. Sixteen out of the 24 calculated effect sizes show an improvement above 10 percent, while six show an improvement in the range -10 to 10 percent. It is worth noting that three of the four highest effect sizes are associated with the experiments conducted at Microsoft [38]. Only two effect sizes show an improvement below—10 percent (effectively, a deterioration of quality under TDD)—both corresponding to experiments from [48].

4.2 Subgroup Analyses at Level 1

The aggregate analysis indicates that TDD has a small conducted. The summary values for each of these subgroups prominent in the unstandardized analysis due to the inclusion of a number of industrial studies. However, the validity of this summary effect, and possible generalization about the impact of TDD from it, can be questioned due to the differences in the experiments analyzed. Better conclusions might be reached if the impact of TDD was assessed through experiments with relatively similar conditions, which was the rationale for conducting the forthcoming level 1 and 2 analyses that focus on the comparative analysis of subgroups of the experiment set.

4.2.1 Academic versus Industrial

This subgrouping strategy is based on the context in which the related experiments were conducted. The summary values for each of these subgroups in the standardized analysis and other synthesis-related details are given in Table 4. As only one of the effect sizes refers to an experiment conducted in an industrial context, only the experiments from the Academic subgroup lend themselves to the standardized analysis, which results in the summary effect size of -0.049. Nevertheless, it is interesting to note that the sole industrial experiment by Slyngstad et al. [63] reported a contrasting effect size of 0.309. Due to only a single industrial experiment being included in the standardized analysis, the Level 1 summary value of the Academic subgroup is not very different from the Level 0 summary value of -0.010. The p-value of this subgroup under the random-effects model indicates that there is no difference in quality among the two development approaches.

In this case, the unstandardized analysis may be more useful since it incorporates a larger number of industrial studies. The improvement is calculated for 16 effect sizes from academic experiments and eight from industrial experiments. Under the unstandardized analysis, the experiment effect sizes are shown in Table 6, whereas the summary values of the two subgroups are shown in Table 5. The histogram resulting from this subgrouping strategy is shown in Fig. 3a. Interestingly enough, the summary effect size changes from the Level 0 value of 24 percent: It drops to

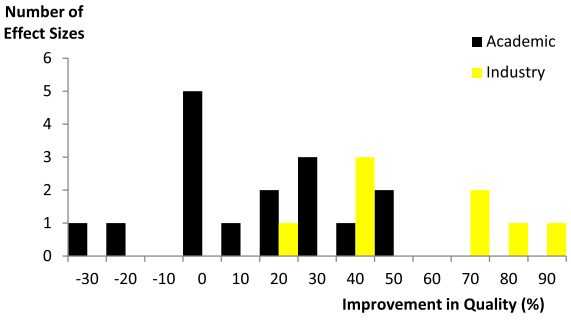
TABLE 5
Quality: Summary of Unstandardized Analysis

level	type	# of effect sizes	% improvement
0	OVERALL	24	24
1	Academic	16	10
	Industrial	8	52
	Waterfall	14	28
	ITL	10	19
	Agile-Inclusive	6	4
	Agile-Exclusive	18	30
2	Academic+Waterfall	10	24
	Academic+ITL	6	-13
	Industrial+Waterfall	4	39
	Industrial+ITL	4	65

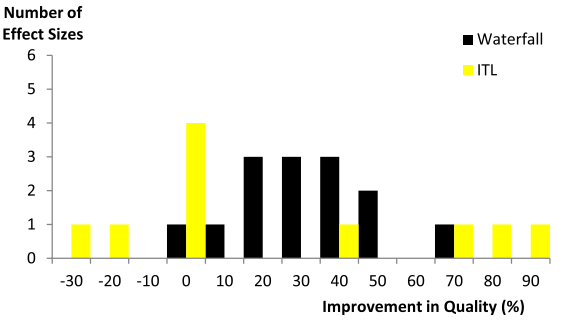
2. For unstandardized analyses, the effect size is reported as a percentage so as to distinguish it from the values obtained in the corresponding standardized analyses.

TABLE 6
Quality: Effect Sizes

experiment	% improvement
Madeyski E2 [48]	-35
Madeyski E1 [48]	-27
Gupta & Jalote E2 [47]	-7
Pančur & Ciglarič E2, E4 [49]	-6
Pančur *et al.* [55]	-3
Huang & Holcombe [52]	-3
Erdogmus *et al.* [5]	-2
Desai *et al.* [50]	3
Gupta & Jalote E1 [47]	16
George E1 [28]	16
George E2 [28]	18
Rahman [56]	24
Vu *et al.* [57]	28
Zhang *et al.* [60]	28
Dogša & Batič [61]	33
Slyngstad *et al.* [63]	33
Yenduri & Perkins [59]	35
Williams et al [30]	39
Edwards [32]	46
Xu & Li [58]	49
Nagappan *et al.* E1 [38]	62
Lui & Chan [62]	67
Nagappan *et al.* E2 [38]	76
Nagappan *et al.* E3 [38]	90



(a) Distribution by Academic/Industry subgroups



(b) Distribution by Waterfall/ITL subgroups

Fig. 3. Distribution of improvement in quality.

10 percent for the academic subgroup (where mixed results are observed), but rises to 52 percent for the industrial subgroup, with a minimum improvement of 18 percent.

4.2.2 Waterfall versus ITL

This subgrouping strategy is based on the testing process utilized by the control group in the experiments: Waterfall or ITL. The summary values for each of these subgroups in the standardized analysis and other synthesis-related details are given in Table 4. In the standardized analysis, the summary effect size rises to 0.301 for the Waterfall subgroup, which corresponds to a small positive improvement according to the classification scheme from [45]. However, the p-value indicates that this result is not statistically significant. The corresponding summary effect for the ITL subgroup drops to -0.403, indicating a medium drop in quality, which is statistically significant, unlike the result of the Waterfall subgroup. It is worth noting that all effect sizes in the ITL subgroup indicate a negative effect of TDD.

The unstandardized analysis includes 14 effect sizes from Waterfall-based experiments and 10 from ITL-based experiments. The summary values of the two subgroups are shown in Table 5 and a histogram based on this subgrouping strategy is shown in Fig. 3a. As shown, the summary value rises to 28 percent for the Waterfall subgroup, but drops to 19 percent for the ITL subgroup. Nevertheless, the histogram shows that we can't simply conclude that TDD performs better than Waterfall, but worse than ITL. As can be seen, the results relating to the ITL subgroup are rather sharply divided, with about two-thirds of the effect sizes being below those in the Waterfall subgroup and the remaining third being higher than those in the Waterfall subgroup. A deeper analysis of this observation is conducted as part of the Level 2 unstandardized subgroup analyses below.

4.2.3 Agile-Inclusive versus Agile-Exclusive

This subgrouping strategy is based on the presence of other agile practices (Agile-Inclusive) or lack thereof (Agile-Exclusive).

The summary values for each of these subgroups in the standardized analysis and other synthesis-related details are given in Table 4. In the standardized analysis, only two of the analyzed experiments belong in the Agile-Inclusive subgroup. As a result, not much can be derived from using this subgrouping strategy in the standardized analysis. In the Agile-Inclusive subgroup, a summary effect size of -0.385 is computed for the two experiments. The summary value of the Agile-Exclusive subgroup is 0.073, but this result is not statistically significant, similar to the corresponding value in Level 0 standardized analysis.

In the unstandardized analysis, six effect sizes refer to experiments that are part of the Agile-Inclusive subgroup. The summary values of the two subgroups are shown in Table 5. As shown, the Level 0 summary value drops to 4 percent for this subgroup, but rises slightly to 30 percent for the Agile-Exclusive subgroup, exhibiting a pattern similar to that of the standardized analysis.

4.3 Subgroup Analyses at Level 2

In an attempt to explain the variation in effect sizes within a subgroup and perhaps strengthen conclusions obtained through subgrouping the effect sizes at Level 1, we subdivided the effect sizes in each of the Academic and Industrial subgroups at Level 1 according to the use of a Waterfall- or ITL-based development process by the control group.

4.3.1 Academic+Waterfall versus Academic+ITL

In this section, we examine the impact of dividing the effect sizes in the Academic subgroup.

The summary values for each of these Level 2 subgroups in the standardized analysis are given in Table 4. In the standardized analysis, the summary effect size value increases to 0.285 for the Academic+Waterfall subgroup, but decreases to -0.403 for the Academic+ITL subgroup. The conclusion is thus similar to that obtained at Level 1, Waterfall-versus-ITL, standardized analysis reported above—in both cases the performance of TDD is superior in the Waterfall-based subgroup. At the same time, similar to the Level 1 standardized analysis, only the result pertaining to the Academic+ITL subgroup is statistically significant.

The summary values for each of these Level 2 subgroups in the unstandardized analysis are given in Table 5. Similar behavior is observed in the unstandardized analysis, where the summary value for the Academic+Waterfall subgroup rises to 24 percent, whereas that for the Academic+ITL subgroup drops to -13 percent.

4.3.2 Industrial+Waterfall versus Industrial+ITL

Standardized analysis makes no sense for this subgrouping at Level 2 as there is only one effect size related to Industrial experiments in this analysis.

The summary values for each of these Level 2 subgroups in the unstandardized analysis are given in Table 5. In the unstandardized analysis, the Level 1 summary value of the industrial subgroup drops for the Waterfall subgroup, but rises for the ITL subgroup: Both the summary values of 39 and 65 percent are in sharp contrast to the result of the Level 2 analysis of the Academic subgroup. This result may be used to provide additional insight into the observation made with respect to the histogram in Fig. 3b. Namely, it turns out that the two-thirds of the effect sizes in the ITL subgroup that were lower than the Waterfall subgroup at Level 1 originate from the experiments conducted in an academic context; the remaining one-third of the effect sizes that were higher than those from the Waterfall subgroup originate from the experiments conducted in an industrial context.

4.4 Discussion

Let us now try to explain where differences between the effect sizes of the subgroups obtained under the different subgrouping strategies come from.

4.4.1 Academic versus Industrial

In the unstandardized analysis, a clear difference was observed in the summary effect sizes for the Academic and Industrial subgroups, with the latter resulting in larger improvement in quality. These differences are likely caused by the differences in developer experience and task size, as can be ascertained from the information provided in Tables 2 and 3.

With regard to developer experience, it is quite plausible that industrial developers, who can be reasonably expected to have much higher experience level than the academic ones, are likely to achieve higher TDD conformance. In this manner, industrial developers can utilize the TDD technique to its full potential, and thus be able to obtain net quality improvement reported in industrial experiments.

With regard to task size, empirical research on TDD has found that the benefits of TDD might not be immediately apparent and may require some time before progressively becoming visible [64]. As the projects that were implemented in the academic studies were generally much smaller in size than those implemented in the industrial studies, the resulting improvements in quality simply have not had enough time to materialize, thus resulting in summary effect sizes that were comparatively smaller than those in the industrial studies, or even nonexistent in some cases.

4.4.2 Waterfall versus ITL

Both standardized and unstandardized analyses found noticeable differences in effect sizes between the Waterfall and ITL subgroups. To explain these differences, let us note first that the primary difference between the Waterfall and ITL subgroups lies in the amount and distribution of test effort during the control group's development process, as explained above.

The standardized analysis, primarily illustrating results from Academic studies, found that the larger improvements of effect sizes were generally obtained in the Waterfall subgroup, while all of the effect sizes were negative in the ITL subgroup. This difference can be explained by the difference in test effort, but only in part. Namely, it is highly plausible that a large difference in test effort (as reported in the Waterfall subgroup) led to larger quality improvements observed. However, when the test effort in one group is roughly equivalent to that in the other group, as is the case with the experiments in the Academic+ITL subgroup, the summary effect size actually indicates a drop in quality. Obviously other interacting forces or variables exist that are not immediately known from the results of this analysis.

The unstandardized analysis found that the effect sizes in the Waterfall subgroup were higher than about two-thirds of effect sizes in the ITL subgroup, but lower than the remaining third. This difference cannot be justified solely by the test effort. In particular, effect sizes related to experiments in the Industrial+ITL subgroup are generally higher despite having much smaller or negligible improvements in test effort, which is counterintuitive.

To reduce the number of the confounding variables, we have conducted Level 2 comparison of Industrial+Waterfall versus Industrial+ITL subgroups, only to find that effect sizes were still generally higher in the ITL subgroup. Upon careful inspection of the latter subgroup, it was noticed that three (out of four) effect sizes originate from the same study [38]. All three experiments employed a modified version of TDD which included an initial detailed design phase which may well have led to larger improvements in quality. If the three experiments are excluded, the sole remaining experiment in the Industrial+ITL group [61] shows improvement of 33 percent, much closer to the summary effect size of 39 percent for the Industrial+Waterfall subgroup. This finding may be interpreted to mean that at large task sizes, as are those encountered in industry, an improvement in quality is less dependent of the level of test effort as it is on the actual development process used. However, this result cannot be confirmed due to the small number of studies involved, and should be verified by future empirical research.

TABLE 7
Productivity: Summary of Standardized Analysis

level	type	# of Effect Sizes	Model	Hedges g	95% CI	p	Q	I^2	df
0	OVERALL	10	fixed-effects	0.064	-0.195 . . 0.324	0.628	41.104	78.104	9
			random-effects	0.048	-0.522 . . 0.618	0.869	11.912	24.448	9
1	Academic	9	fixed-effects	0.212	-0.063 . . 0.487	0.131	31.181	74.343	8
			random-effects	0.187	-0.376 . . 0.749	0.515	11.253	28.906	8
	Industrial	1	fixed-effects	-1.111	-1.887 . . -0.335	0.005	0.000	0.000	0
			random-effects	-1.111	-1.887 . . -0.335	0.005	0.000	0.000	0
	Waterfall	5	fixed-effects	-0.536	-0.954 . . -0.119	0.012	18.988	78.934	4
			random-effects	-0.436	-1.385 . . 0.514	0.369	6.196	35.447	4
	ITL	5	fixed-effects	0.443	0.112 . . 0.774	0.009	9.127	56.174	4
			random-effects	0.465	-0.043 . . 0.974	0.073	4.005	0.136	4
	Agile Inclusive	3	fixed-effects	0.620	0.208 . . 1.033	0.003	6.872	70.897	2
			random-effects	0.702	-0.089 . . 1.493	0.082	1.862	0.000	2
	Agile Exclusive	7	fixed-effects	-0.299	-0.633 . . 0.034	0.079	22.694	73.561	6
			random-effects	-0.273	-0.942 . . 0.395	0.423	8.718	31.178	6
2	Academic+Waterfall	4	fixed-effects	-0.303	-0.798 . . 0.192	0.230	16.025	81.279	3
			random-effects	-0.230	-1.440 . . 0.979	0.709	4.616	35.007	3
	Academic+ITL	5	fixed-effects	0.443	0.112 . . 0.774	0.009	9.127	56.174	4
			random-effects	0.465	-0.043 . . 0.974	0.073	4.005	0.136	4
	Industrial+Waterfall	1	fixed-effects	-1.111	-1.887 . . -0.335	0.005	0.000	0.000	0
			random-effects	-1.111	-1.887 . . -0.335	0.005	0.000	0.000	0
	Industrial+ITL	0	n/a						

4.4.3 Agile-Inclusive versus Agile-Exclusive

The unstandardized analysis shows that inclusion of other agile practices, as exhibited by the effect sizes of the Agile-Inclusive subgroup, has a detrimental impact on quality. However, this drop may well be due to the influence of variables such as experience, task size, and use of an ITL-based development process in the control group that overshadowed the influence of other agile practices. The observation that two-thirds of the experiments in the Agile-Inclusive subgroup were conducted in an academic context, and also that two-thirds of the experiments employed an ITL-based development process for the control group may be interpreted as an indication of this. Still, no conclusive evidence can be identified regarding the impact of other agile practices on the quality obtained through TDD.

5 Results on Productivity

5.1 Aggregate Analyses

5.1.1 Standardized Analysis

A summary of the standardized analysis on the productivity construct is given in Table 7. A total of 10 effect sizes were computed for 12 experiments which employed a total of 265 subjects. Summary effect sizes of 0.064 and 0.048 were computed under the fixed-effects and random-effects models, respectively; both values indicate a very small to negligible impact on productivity. A similar result is found by examining the p-values which indicate that there is no significant difference in productivity between the two development approaches. However, this result should not be generalized as there are many variables that could affect TDD performance, as discussed in Section 3.3. Heterogeneity, as shown by the value of I^2, is high in the fixed-effects model. In view of the differences in rigor and the high heterogeneity level, similar to the quality construct, a random-effects model is adopted for the standardized analysis.

The forest plot for the analysis on productivity under the random-effects model is shown in Fig. 4. As can be seen, four of the effect sizes are positive, with three values being of medium to high magnitude according to the classification in [45]. The remaining six values are negative; three of these have a magnitude of medium or higher. However, p-values indicate that there is no significant difference in productivity in 5 out of 10 experiments.

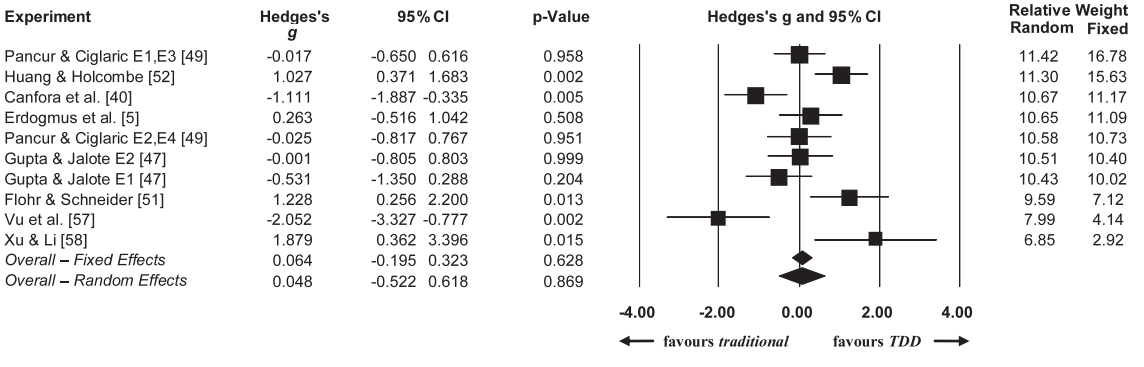


Fig. 4. Forest plot on productivity under the random-effects model.

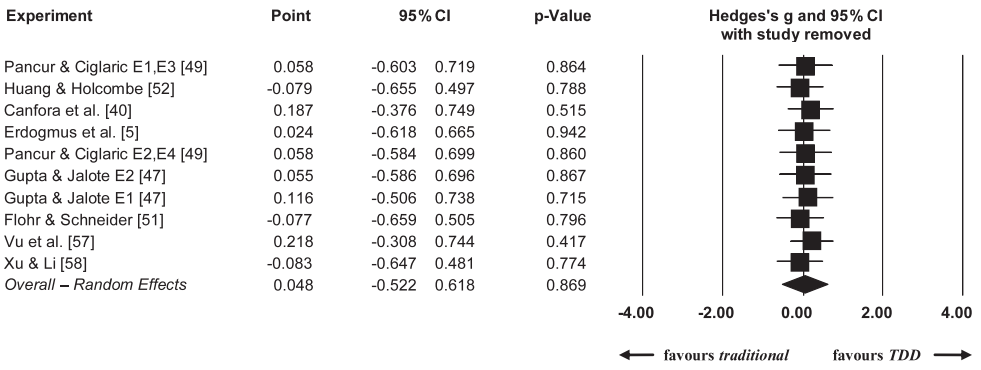


Fig. 5. One-study removed plot on productivity under the random-effects model.

The one-study removed plot for the analysis on productivity is shown in Fig. 5. The summary effect size is most sensitive to the experiments by Vu et al. [57], Canfora et al. [40], and Gupta and Jalote E1 [47].

5.1.2 Unstandardized Analysis

Unstandardized analysis included 23 effect sizes from 27 experiments that employed a total of 564 subjects. As in the case of the quality construct, unstandardized analysis differs from the standardized one in that the results from a much larger number of Industrial experiments are included. A summary of the unstandardized analysis on productivity is given in Table 8, while the individual effect sizes are given in Table 9. The mean improvement is 4 percent, which supports the finding of the standardized analysis, namely, that the impact of TDD on productivity is very small to negligible. Eleven effect sizes are greater than 10 percent, while eight of them are lower than -10 percent; the remaining four individual effect sizes lie in the range -10 to 10 percent. Interestingly enough, two of the experiments reported in [38] reported significant drops in productivity.

5.2 Subgroup Analyses at Level 1

We have also undertaken analysis by subgroups at Level 1, using a classification analogous to that reported in Section 4.2.

5.2.1 Academic versus Industrial

Similarly to the standardized analysis on quality, only one industrial experiment could be included in the standardized analysis on productivity, namely, the experiment by Canfora et al. Hence the results analyzed in the standardized analysis primarily portray the performance of TDD within the academic context. The summary effect size value thus increases from 0.048 at Level 0 to 0.187 at Level 1. Like its counterpart at Level 0, the summary result of the Academic subgroup is not statistically significant.

Unstandardized analysis includes 15 effect sizes from academic experiments and eight from industrial ones, as shown in the histogram of the improvement distribution in Fig. 6a. The summary improvement rises to 19 percent for the academic subgroup, with 12 out of 15 results being above 10 percent—but drops to -22 percent for the industrial subgroup in which seven out of eight effect sizes were below -10 percent.

5.2.2 Waterfall versus ITL

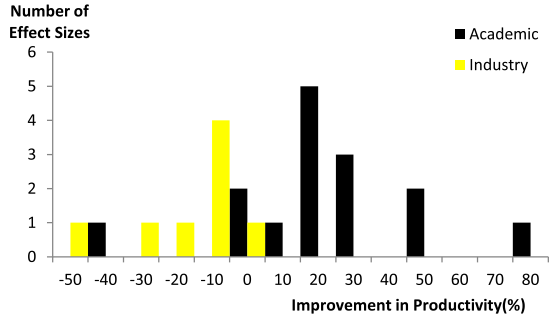
This subgrouping strategy results in more balanced subgroup sizes in the standardized analysis. The summary effect size drops to -0.436 for the Waterfall subgroup, but rises to 0.465 for the ITL subgroup; the latter results are statistically significant.

TABLE 8
Productivity: Summary of Unstandardized Analysis

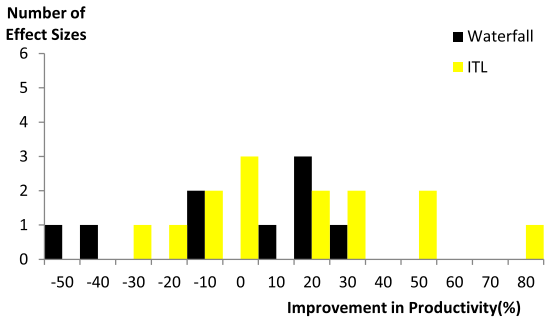
level	type	# of effect sizes	% improvement
0	OVERALL	23	4
1	Academic	15	19
	Industrial	8	-22
	Waterfall	9	-6
	ITL	14	11
	Agile-Inclusive	5	12
	Agile-Exclusive	18	2
2	Academic+waterfall	6	7
	Academic+ITL	9	27
	Industrial+waterfall	3	-30
	Industrial+ITL	5	-17

TABLE 9
Productivity: Effect Sizes

experiment	% improvement
Canfora *et al.* [40]	-57
Xu & Li [58]	-49
Nagappan *et al.* E1 [38]	-30
Nagappan *et al.* E3 [38]	-23
Williams *et al.* [30]	-18
Dogša & Batič [61]	-16
George E2 [28]	-16
Nagappan *et al.* E2 [38]	-15
Pančur & Ciglarič E2, E4 [49]	-5
Pančur & Ciglarič E1, E3 [49]	-1
Geras *et al.* [35]	0
Zhang *et al.* [60]	10
Janzen E2 [42]	10.5
Janzen E3, E4 [42]	13
Gupta & Jalote E2 [47]	14
Vu *et al.* [57]	19
Gupta & Jalote E1 [47]	20
Flohr & Schneider [51]	21
Yenduri & Perkins [59]	25
Erdogmus *et al.* [5]	28
Janzen E1, E5 [42]	50
Kaufmann & Janzen [53]	50
Huang & Holcombe [52]	72



(a) Distribution by Academic/Industry subgroups



(b) Distribution by Waterfall/ITL subgroups

Fig. 6. Distribution of improvement in productivity.

In the unstandardized analysis the Waterfall and ITL subgroups included 9 and 14 effect sizes, respectively, with summary values of -6 percent for the Waterfall subgroup and 11 percent for the ITL subgroup (the corresponding Level 0 value was 4 percent). The histogram of the improvement distribution, shown in Fig. 6b, illustrates that the ITL subgroup had better productivity in general, despite the fact that some studies using ITL performed worse than some of the Waterfall-based ones.

5.2.3 Agile-Inclusive versus Agile-Exclusive

In the standardized analysis, only three effect sizes refer to experiments that are part of the Agile-Inclusive subgroup; the other Agile-Exclusive subgroup included seven effect sizes. The summary effect size increased substantially to 0.702 for the Agile-Inclusive subgroup (which is statistically significant) and decreased to -0.273 for the Agile-Exclusive subgroup. However, small subgroup sizes make these results hard to generalize.

In the unstandardized analysis, the Agile-Inclusive and Agile-Exclusive subgroups included five and 18 effect sizes, respectively, with summary value of 12 percent for the Agile-Inclusive subgroup and 2 percent for the Agile-Exclusive subgroup. We note that the increase in summary value upon inclusion of other agile practices has been observed in both the standardized and unstandardized analyses of quality.

5.3 Subgroup Analyses at Level 2

We have also undertaken analysis by subgroups at Level 2, using a classification analogous to that reported in Section 4.3.

5.3.1 Academic+Waterfall versus Academic+ITL

Since the standardized analysis includes data from a single industrial experiment, the result of this Level 2 analysis is expected to be very similar to the result of the Level 1 analysis under the Waterfall versus ITL subgrouping strategy. Indeed, the resulting summary value was -0.303 for the Academic+Waterfall subgroup and 0.443 for the Academic+ITL subgroup. Since the same experiments comprise the ITL-based subgroup in both the Level 1 and 2 analyses, the result of the ITL subgroup in the Level 2 analysis is also statistically significant. In the Level 2 unstandardized analysis, the summary value drops to 7 percent for the Academic+Waterfall subgroup and rises to 27 percent for the Academic+ITL subgroup. Both standardized and unstandardized analyses indicate that the use of ITL seems to cause an improvement in the summary effect size.

5.3.2 Industrial+Waterfall versus Industrial+ITL

At Level 2 no results are available from the standardized analysis as there is a single industrial experiment that can be analyzed.

In the unstandardized analysis, we obtained summary values of -30 percent for the Industrial+Waterfall subgroup and -17 percent for the Industrial+ITL subgroup. This result is similar to that observed in the Level 2 analysis of the Academic subgroup, where the use of ITL has led to an increase in summary effect size.

The results of the last two Level 2 analyses indicate that in both academic and industrial experiments, larger improvements in productivity were noted for the ITL-based subgroups, despite the apparent interleaving of the effect sizes of individual experiments.

5.4 Discussion

With the results above, we can attempt to rationalize the differences in the effect sizes of subgroups under the various subgrouping strategies. But before that, it is worth noting that two contradictory perspectives were described in the literature regarding the impact of TDD on productivity as summarized by Turhan et al. [65]. One viewpoint suggests that TDD results in improved productivity by improving external code quality, internal code quality, and test quality, all of which lead to lower defect generation rate, faster rate of defect detection and fixing, and easier and faster maintenance. The opposing viewpoint is that the TDD incurs increased testing overhead, which degrades productivity. Let us now discuss the results of subgroup analyses and try to determine which of these two opposing viewpoints, if any, can justify our results.

5.4.1 Academic versus Industrial

The unstandardized analysis demonstrates a noticeable difference between the effect sizes of the Academic and Industrial subgroups, the former being generally higher than the latter. Moreover, none of the experiments in the industrial subgroup indicated an improvement in productivity. As mentioned in the discussion on quality in Section 4.4, the two subgroups substantially differ in terms of size and task developer experience, which may well lead to the differences between the effect sizes in the two subgroups.

The impact of task size on productivity, if any, cannot be inferred from the details available on the experiments included in this analysis.

As for developer experience, many among the academic studies have reported problems with process conformance in the treatment groups as a significant number of subjects were unable to grasp the TDD style, test the code properly, and respect other practices that comprise TDD. It may be argued that the developers with longer experience are able to achieve higher levels of process conformance and thus pay more attention to, and spend more time in, TDD-specific activities such as unit testing, refactoring, and the like, which increases project duration and decreases productivity. On the other hand, unstandardized analysis on productivity unequivocally shows that larger drops in productivity occur in the industrial experiments, even though these same experiments report larger quality improvements.

Together, these two observations seems to support the viewpoint that proper application of TDD incurs a larger overhead and thus degrades productivity.

5.4.2 Waterfall versus ITL

Both standardized and unstandardized analyses found larger productivity improvements in the ITL subgroup than in the Waterfall subgroup. Moreover, our Level 2 analysis has shown that 1) all of the effect sizes from industrial experiments were smaller than all of the academic experiments, and 2) in each of these two sets of experiments, the ITL-based experiments experienced higher productivity improvements than the Waterfall-based ones. The primary difference between the ITL- and Waterfall-based process is the amount of test effort, as explained in Section 3.3. In scenarios where the difference in test effort (when switching to TDD) was not substantial, i.e., in the ITL subgroup, productivity improvements were typically higher than in the scenarios from the Waterfall subgroup where the difference in test effort was significant.

Therefore, we may conclude that most, if not all, of the difference in effect sizes is likely due to the difference in test effort. This conclusion again supports the viewpoint that the application of TDD incurs a larger testing overhead and thus results in a drop in productivity. At the same time, the switch from test-last (as in ITL) to test-first (as in TDD) development did improve productivity, most likely due to reduced defect generation and improved defect detection and correction offered by TDD as stipulated by the other viewpoint.

5.4.3 Agile-Inclusive versus Agile-Exclusive

Unstandardized analysis has shown that larger productivity improvements were experienced in the experiments which incorporated agile practices other than TDD, primarily pair programming. One possible explanation for this result could be that TDD leads to good synergy with other agile practices and thus including them improves the performance of TDD-based development. However, an alternate justification for our results is also possible. Namely, all but one of the studies in the Agile-Inclusive subgroup employed a conventional but ITL-based development process. Therefore, the productivity improvements tentatively associated with other agile practices might, in fact, be due not to other agile practices, but to other factors as yet unaccounted for in our analysis. (One such factor could be the lower defect generation and improved defect detection and correction rates noted in the previous paragraph.) In light of these claims, any judgments on the impact of other agile practices are reserved until future research.

6 Moderator Variables

The above discussion on the outcome constructs has shown that differences in effect sizes of subgroups are most likely due to other variables. Two among the most important variables are developer experience and task size, as highlighted in the discussions in the Academic versus Industrial subgrouping. In this section, we take a closer look at the moderating effects of these variables, beginning with a brief summary of previous research that documents the impact of these variables on performance of TDD-based development process.

6.1 Developer Experience

Although the success of TDD is dependent on skills in a number of different areas, including programming, testing, design, refactoring, and thinking in a TDD style [2], [3], programming experience and exposure to TDD are the only variables that have been explicitly studied. Table 10 summarizes existing studies that relate the impact of experience on TDD.

Müller and Höfer [66] compared the performance of final-year undergraduate students from an XP course with that of professionals with at least five years of industrial programming experience. The latter group was also more experienced with regard to use of automated testing tools and exposure to TDD. All subjects individually developed a Java-based elevator control system following the TDD approach until they felt they were done; their programs were then evaluated using previously prepared acceptance tests. The subjects in the professional group were found to finish the task in shorter time, and this result was statistically significant. The difference was attributed to faster coding speed and higher level of programming experience. However, a larger proportion of programs prepared by the students passed the acceptance tests, but this result was not statistically significant. This somewhat unexpected result was attributed to the professionals' perception of the acceptance testing process as a regular adjunct to testing and thus being assigned lower priority. On the other hand, subjects in the students group viewed the acceptance tests as a more formal assessment criteria, and thus ensured, to a greater degree, that the implemented functionality was in working order prior to submission.

Müller and Höfer's experiment was repeated by Höfer and Philipp [67] using the same experimental task but with two distinct student/novice groups in which subjects worked in pairs. Subjects in the novice groups were students who took the same XP course as [66] in later years, with one group being selected from the 2006 class and the other from the 2008 class. Students in the novice groups differed in programming experience, but they had similar levels of expertise in TDD and the JUnit tool. The professional group was mostly comprised of employees of a

TABLE 10
Empirical Studies Investigating the Influence of Developer Experience

study	subjects/pairs type	#	context	experience (years) general software dev.	Java/C/C++	testing	TDD	code size (LOC)	duration (hours)	Quality Result	Productivity Result
Müller and Höffer [66] (individual)	Novice	11	XP course 04	6	2.4	NSE [a]	NSE [b]	450	4-5 hrs	35% more of the programs by NG passed AT.	EG took approx. 31% less time than NG (SS). [c]
	Expert	7	Multiple Sources	7.0	6.2	4.3 (JUnit)	3.4				
Höfer and Philipp [67] (pairs)	Novice 06	12	XP course 06	4.8	2	NSE [d]	-	450	4-5 hrs	-	EG took approx. 37% more time than NG'08 (SS). EG took approx. 36% more time than NG'06. [c]
	Novice 08	12	XP course 08	6.4	4	NSE [e]	NSE [f]				
	Expert	12	company (see text)	7.8	7.2	5.5 (JUnit)	3.0				
Madeyski [68] (pairs)	2nd year	39	Programming in Java Course		-	-	-	27 user stories	12 hrs	-	-
	3rd year	27									
	4th year	4									
	MSc	70									

Note: All numerical values including those for experience, code size and duration are averages for all subjects in the respective group.
Acronyms - NG: Novice Group, EG: Expert Group, NSE: No Significant Experience, AT: Acceptance Tests, SS: Statistically Significant
[a] 4 subjects had prior exposure to JUnit.
[b] 1 subjects had prior exposure to TDD.
[c] Percentage calculated using median values approximated from a box-plot.
[d] 1 subject had prior exposure to JUnit.
[e] 3 subjects had prior exposure to JUnit.
[f] 3 subjects had prior exposure to TDD.

company specializing in agile software development and consulting. Interestingly, results from this experiment visibly differed from that of the previous one. In particular, subjects in the professional group generally took longer to complete the program than those in either of the novice groups. The difference between the expert group and the novice 2008 group was statistically significant, unlike the one between the professional and the novice 2006 group and the one between the two novice groups. The extra time utilized by the professional group was attributed to longer time spent refactoring code, a finding that was also statistically significant. However, results for quality were not reported in this experiment.

Madeyski [68] analyzed the impact of experience on quality using data from a previously conducted experiment [48]. This experiment has a much larger subject group than the previous two and it aimed to compare the performance of pairs applying the classic testing approach with those applying the TDD approach. The subjects' experience level was determined by the academic year in which they were enrolled, but little other information about the subjects' experience and skill levels was collected. The results indicated a small correlation between programmer experience and quality in the group that used TDD, but this observation was not statistically significant.

As these studies report conflicting results, no definite conclusion can be drawn. Only one of the studies reported experience adequately and had an expert group that plausibly applied TDD to its full potential [67], and its results indicate that productivity was lower for the expert group. This finding is aligned with our findings on productivity in the Academic versus Industrial subgrouping, where industrial developers (arguably analogous to the expert group in said study) were found to perform at lower productivity level than the student ones.

We did attempt to correlate effect size data with the experience-level data, but the classification of studies into subgroups with "low", "medium," and "high" experience level could not be done in a reliable manner due to the lack of data on experience level in the various areas such as testing, refactoring, etc.

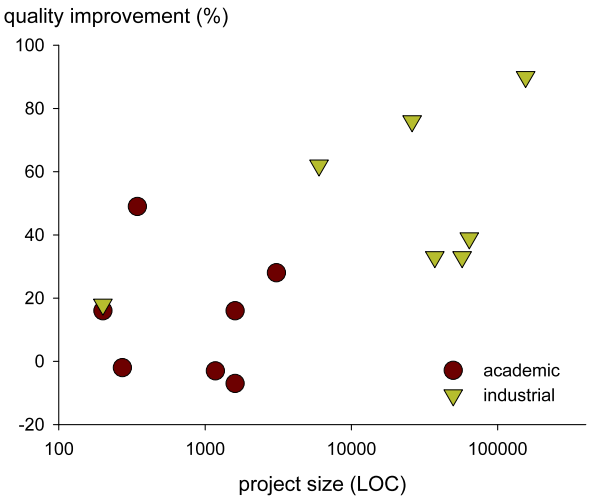
6.2 Task Size

We have also analyzed the relationship between task size and the magnitude of the improvement brought about by TDD; 14 data points (effect sizes and respective task sizes) could be obtained from the analysis on quality and 13 from the analysis on productivity. Fig. 7 shows the scatterplots of percentage improvement measures for both outcome constructs versus task size; for clarity, the x-axis uses a logarithmic scale. A visible trend can be identified in the plot for quality, unlike the plot for productivity where such a trend cannot be easily observed; the relationships of both quality and productivity improvements with task size appear to be logarithmic in nature. Table 11 shows the Spearman correlation coefficients between the percentage improvement and the log transform of task size; as can be seen, the task size was found to be correlated with quality, and this correlation is statistically significant.

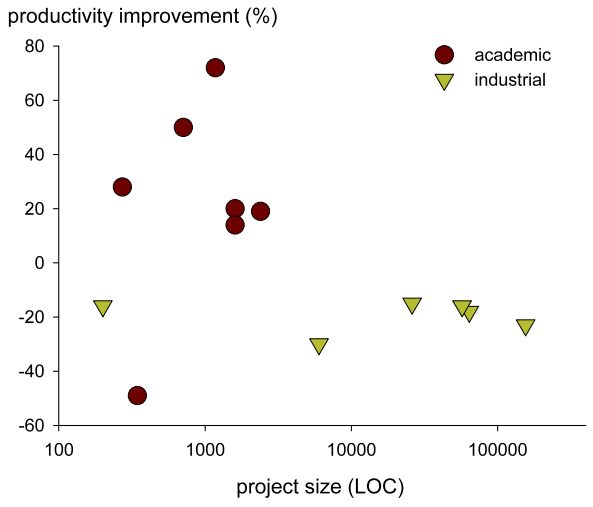
Interestingly enough, no previous empirical research could be found that related task size with performance of a TDD-based development process. However, two studies did hint at the possible influence of this variable: When recalling experiences on the adoption of agile practices in industry, Hodgetts [64] claimed that the implementation of TDD took several two-week XP cycles to show significant results. Also, in a study conducted in an academic environment, Erdogmus et al. [5] explained the lack of improvement in quality by reasoning that when the programming task is small, ad hoc testing strategies, visually inspecting code, and other conventional practices could perhaps substitute for the benefits brought about by TDD, thus making the advantages of TDD more or less transparent.

6.3 Other Variables

In both previous sections, some variation among effect sizes within subgroups still exist, even upon subgrouping



(a) Improvement in quality.



(b) Improvement in productivity.

Fig. 7. Scatterplot showing the relationship of task size to improvements in quality and productivity.

at Level 2. This might be due to the influence of other variables, most notably task complexity. Higher values of task complexity might have a detrimental impact on the magnitude of the quality improvement, as was observed in [47] where two experiments were carried out with the same subjects and similar task size, but with different task complexity levels. On the other hand, higher levels of task complexity may make TDD more difficult to apply, time- and effort-wise, thus leading to a negative overall impact on productivity. However, more detailed investigation into this issue is needed before any definitive conclusions can be drawn.

7 Threats to Validity

The major obstacle in conducting this analysis was the lack of data available for computing the standardized effect size in each experiment. Although we partially overcame this obstacle by using an unstandardized effect size measure, all unstandardized measures within the context of this research suffer from two principal disadvantages.

TABLE 11
Correlation of Task Size with Quality and Productivity

	correlation coefficient	# of Effect Sizes	p-value
quality	0.65	14	0.028
productivity	-0.29	13	0.562

First, since the metrics used to operationalize the outcome constructs differ from study to study, as explained in Section 3.2, differences in these metrics as well as differences in scale may affect the accuracy of the comparison between the results of two studies. It is worth noting that this threat exists in the context of standardized analysis as well.

Second, using means or medians while ignoring the standard deviation within the results of subjects in a group, might result in an incorrect assessment of the actual effect size when comparing the results of two groups a study, hence leading to conclusions that are misleading or exaggerated.

The selection of a parametric effect size measure such as the Hedges' g is based on the assumption that the outcome construct being investigated is normally distributed. If this is not the case, then a nonparametric measure should be used. However, this is not a feasible option as such measures can only be used for characterizing data, but not for relating it back to a population [8]. Due to the lack of availability of raw data that could point to the right distribution and considering that means are meaningful indicators for both of the outcome constructs, a parametric measure was adopted as the best option.

As noted in [43], many among the studies on TDD consider only the initial development phase when calculating productivity, while ignoring any long-term effects on maintenance. Thus it is difficult to get a comprehensive understanding of the long-term impact of TDD on productivity from our analysis, and the conclusions derived should be considered to apply solely to productivity during the initial development phase.

As studies differed in their adherence to the TDD process, it is difficult to estimate the accuracy with which the summary effect sizes calculated in our analysis provide a reflection of the true effectiveness of TDD.

Finally, publication bias [8] is another factor that should be considered when a meta-analysis is undertaken, but the data in the studies included here—especially those in the industrial group—are insufficient to make a meaningful analysis of this kind.

8 Comparison with Earlier Reviews

Our work is not the first to attempt to summarize research on the efficacy of TDD. In fact, Siniaalto [69], Kollanus [70], and Turhan et al. [65] published review studies with the same goal. Table 12 shows the findings of these review studies and lists the studies from which the original data on external quality and productivity outcome constructs (studies that reported on internal and design quality are omitted). (We also note that Shull et al. [43] summarized the findings of [65] and contrasted its results with expert opinion.) A brief summary of the methodology and results of the three earlier reviews is presented next.

TABLE 12
Comparison with Previous Reviews

review	year	studies included	External Quality	Productivity
Siniaalto [69]	2006	[5] [13] [14] [15] [25] [29] [30] [32] [33] [35] [36] [55] [53] [54] [62]	overall: improvement	overall: inconclusive
			academic: inconclusive	academic: inconclusive
			semi-industrial: inconclusive	semi-industrial: inconclusive
			industrial: inconclusive	industrial: improvement
Kollanus [70]	2010	[5] [16] [22] [24] [26] [27] [29] [30] [31] [32] [33] [35] [37] [38] [39] [40] [41] [47] [48] [55] [50] [51] [52] [54] [56] [57] [58] [60] [62] [63] [71]	overall :improvement	overall: degradation
			controlled experiment: no difference	controlled experiment: inconclusive
			case studies: improvement	case studies: degradation
			other: improvement	other: improvement
Turhan *et al.* [65]	2010	[5] [17] [21] [26] [28] [30] [34] [35] [38] [40] [42] [47] [48] [55] [51] [52] [53] [54] [57] [59] [60] [63]	overall: improvement	overall: inconclusive
			controlled experiment: inconclusive	controlled experiment: improvement
			pilot studies: improvement	pilot studies: inconclusive
			industrial: improvement	industrial: degradation
		Note: a number of studies were classified as high-rigor studies; the results were inconclusive across all categories.		
Our review	2011	[5] [28] [30] [32] [35] [38] [40] [42] [47] [48] [49] [50] [55] [51] [52] [53] [54] [56] [57] [58] [59] [60] [61] [62] [63]	overall: improvement	overall: inconclusive
			academic: no difference	academic: improvement
			industrial: improvement	industrial: degradation
			waterfall: improvement	waterfall: degradation
			ITL: inconclusive (potential degradation)	ITL: inconclusive (potential improvement)

Siniaalto [69] examined the findings of 15 studies, categorized into Academic (A), Semi-Industrial (SI), and Industrial (I) contexts. This review was conducted much earlier than the other two and thus includes fewer studies than the other two, which puts it in a less favorable position for obtaining definite conclusions. Overall, TDD was found to result in an improvement in external quality, but the results on productivity are contradictory. In the subgroup analysis of external quality, consistent improvements were found among the industrial studies, while the other two groups provided improvements varying between significant to small or nonexistent. In the subgroup analysis on productivity, results from studies in the academic and industrial subgroups ranged from no difference in productivity to a negative impact, whereas studies from the semi-industrial subgroup reported a nonnegative impact. However, the validity and applicability of results must be interpreted in light of the observation that in a substantial number of studies with student subjects, task sizes were small, different measurement metrics were used, and process conformance was not monitored; hence the results could have been confounded with the impact of other agile practices such as pair programming.

Kollanus [70] conducted a review of 31 studies on the quality and productivity constructs, categorized on the basis of their experimental design into the Controlled Experiments (CE), Case Studies (CS), and Other (O) subgroups. However, conclusions were largely based on the results of the controlled experiments and case studies subgroups. With regard to the external quality construct, Kollanus claims that the empirical evidence indicates "weak" support for an improvement in quality. The weakness arises from the results in the controlled experiments subgroup being mixed, with roughly a third of the studies having reported an improvement in quality. In contrast, almost all of the case studies were found to report an improvement in external quality. With regard to the productivity construct, "moderate" support was found for an increase in development time. A majority of the case studies were found to report a drop in productivity, whereas the results from controlled experiments were mixed, with approximately half of the studies reporting a drop in productivity. The review also observes that in several empirical studies the improvement in quality coincided with a drop in productivity. Following the intuition given in one of the analyzed studies [52], it was hypothesized that an improvement in quality and the accompanying drop in productivity could be the result of increased unit testing rather than an intrinsic trait of TDD. Also, common traits that could affect review validity include a poorly defined control group process, varying developer experience, and small task size.

Turhan et al. [65] conducted a review of 22 studies. Similar to [70], the studies were divided into the Controlled Experiments (CE), Pilot Studies (PS), and Industrial (I) subgroups. Additionally, all studies were classified as being either high rigor or low rigor, according to developer experience, process definition and conformance, and the scale of the study (determined by the number of subjects and task size). Overall, the empirical evidence suggests an improvement in quality, but the results for productivity are mixed. In the subgroup analysis on external quality, the larger share of the industrial-use and pilot studies reported a positive effect on quality, while the results from controlled experiments were inconclusive. However, when considering only high-rigor studies, the supporting evidence for an improvement in external quality disappeared as the results were roughly mixed in each subgroup, thus rendering the overall result inconclusive. With regard to the productivity construct, the overall result was inconclusive, as studies in the controlled experiments and industrial subgroups reported an improvement and a drop in productivity, respectively, whereas results from the pilot studies subgroup were mixed. Turhan et al. also highlighted the two opposing lines of argument held with regard to the impact of TDD on productivity, which are mentioned in the discussion on productivity above.

The approach presented in this paper differs from the previous reviews in a number of important aspects.

1. First, focus is placed on externally observable variables, namely, external quality (represented by defect density) and developer productivity, both of which are directly measurable. Internal code quality is not considered as there is no widely accepted measure for it, while test coverage is not considered as studies differ in the type of coverage (line, branch, method, ...) they report.
2. Second, a quantitative approach was adopted to measure the magnitude of the improvement/drop in each investigated empirical study. This permits performance comparison among the studies and allows us to gauge the size of the improvement from applying TDD, thus making conclusions from the review more meaningful.
3. Third, two major subgrouping strategies were applied: based on the experimental context (i.e., academic/industrial) and based on the control group's development process (Waterfall/ITL). Together with the previous point, this approach allows us to determine reasons that affect the performance in empirical studies.
4. Fourth, potential moderator variables derived from the discussion of results were assessed to determine the extent of their impact on the magnitude of the improvement.
5. Finally, a thorough analysis of rigor was conducted to assess the universal applicability of the available empirical research on TDD.

9 Conclusion

This research intended to investigate the effectiveness of TDD by applying meta-analytical techniques to previous empirical research on the external quality and productivity outcome constructs. Despite considerable differences among the experiments, valuable insight can be gained from this analysis. Overall, our analysis suggests that TDD results in a small improvement in quality but results on productivity are inconclusive. To gain deeper insight into the differing levels of improvement observed for both outcome constructs, we have also conducted subgroup analyses using two major subgrouping strategies.

Under the Academic versus Industrial subgrouping, much larger improvements in quality were found in the Industrial experiments, which may be attributed to higher developer experience and much larger task sizes in those studies. Although the analysis on moderator variables identified a correlation with task size, no concrete evidence was found relating the experience level to the magnitude of the improvement in quality.

Under the Waterfall versus ITL subgrouping, academic experiments that employed a Waterfall-based testing process for the control group reported larger improvements in quality; this may be attributed to a significant increase in test effort in the case of Waterfall process. Among the industrial experiments, both subgroups achieved a similar level of quality improvement, which seems to indicate that the impact of the difference in test effort is not as pronounced at large task sizes. Given that only one experiment in our analysis was both conducted in an industrial context and utilized an ITL-based conventional development process, the conclusion regarding industrial experiments needs to be verified in future research.

Under the Academic versus Industrial subgrouping, larger drops in productivity were generally observed in the Industrial subgroup. This was rationalized in view of the impact of higher developer experience: namely, that more experienced developers put more emphasis and, consequently, devote more time to TDD-specific activities such as testing and refactoring. Although no correlation was found between the task size and the magnitude of the improvement, it seems plausible that extensions or delays in project duration occur in proportion to the project size: small delays at small task sizes, larger delays at large task sizes. Under the Waterfall versus ITL subgrouping strategy, in both the academic and industrial experiments larger drops in productivity were found in the experiments that utilized a Waterfall-based development process, which may be attributed to a larger increase in test effort in their respective experiments.

The internal or design quality construct and its impact on external quality and productivity is a promising topic for future research; however, some conditions need to be satisfied. First, the empirical setting should mandate the use of an ITL-based traditional development process so that the influence of extra test effort can be eliminated. Second, the task size should be large so that any benefits in internal quality and, by extension, on external quality and/or productivity become visible. Third, and arguably the most difficult condition to achieve, is the use of a common metric for internal quality.

As some experiments that coupled TDD with an initial detailed design phase achieved high-quality improvements, the impact of this practice on TDD should be investigated as well, together with the impact of task complexity on the chosen outcome constructs. Again, a common metric for task complexity is a prerequisite for this research.

We note a few key issues that the authors of future empirical studies should be aware of when designing their experiments. Adequate training should be given to the subjects so that they can sufficiently grasp the TDD technique prior to the actual experiment. Detailed information about training should be provided as well, including specific areas as well as the duration of the training. Industrial experiments should attempt to record the long-term impacts of TDD, as opposed to the impacts on the initial development phase. More emphasis should be placed on maintaining a high level of process conformance. Finally, task complexity should be reported in a more objective and standardized fashion so that it can be easily compared across studies.

Acknowledgments

The authors would like to thank the reviewers for their insightful and helpful feedback which helped substantially improve the paper. They would like to thank Mina Tadrous from the University of Toronto for his help in interpreting some of the results of their meta-analysis.

References

[1] S. Ambler, "Test-Driven Development Is the Combination of Test First Development and Refactoring," *Dr. Dobbs' Agile Newsletter*, June 2006.

[2] D. Astels, *Test Driven Development: A Practical Guide.* Prentice-Hall, 2003.

[3] K. Beck, *Test-Driven Development: By Example.* Prentice-Hall, 2003.

[4] J. Andrea, "Test-Driven Development: Driving New Standards of Beauty," *Beautiful Testing: Leading Professionals Reveal How They Improve Software,* pp. 181-194, O'Reilly Media, 2009.

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

[6] M. Lipsey and D. Wilson, *Practical Meta-Analysis.* Sage Publications, Inc., 2001.

[7] B. Kitchenham and S. Charters, "Guidelines for Performing Systematic Literature Reviews in Software Engineering," Keele Univ., Technical Report EBSE 2007-001, 2007.

[8] J. Hannay, T. Dybå, E. Arisholm, and D. Sjøberg, "The Effectiveness of Pair Programming: A Meta-Analysis," *Information and Software Technology,* vol. 51, no. 7, pp. 1110-1122, 2009.

[9] T. Huedo-Medina, J. Sánchez-Meca, F. Marín-Martínez, and J. Botella, "Assessing Heterogeneity in Meta-Analysis: *Q* statistic or I^2 Index?" *Psychological Methods,* vol. 11, no. 2, pp. 193-206, 2006.

[10] G. Melnik and F. Maurer, "A Cross-Program Investigation of Students' Perceptions of Agile Methods," *Proc. 27th Int'l Conf. Software Eng.,* pp. 481-488, 2005.

[11] S. Kollanus and V. Isomöttönen, "Understanding TDD in Academic Environment: Experiences from Two Experiments," *Proc. Eighth Int'l Conf. Computing Education Research,* pp. 25-31, 2008.

[12] A. Rendell, "Effective and Pragmatic Test Driven Development," *Proc. AGILE,* pp. 298-303, 2008.

[13] J. Langr, "Evolution of Test and Code via Test-First Design," *Proc. ACM Conf. Object-Oriented Programming, Systems, Languages, and Applications,* http://www.objectmentor.com/resources/articles/tfd.pdf, Oct. 2001.

[14] D.H. Steinberg, "The Effect of Unit Tests on Entry Points, Coupling and Cohesion in an Introductory Java Programming Course," *Proc. XP Universe,* Oct. 2001.

[15] P. Abrahamsson, A. Hanhineva, and J. Jäälinoja, "Improving Business Agility through Technical Solutions: A Case Study on Test-Driven Development in Mobile Software Development," *Proc. Int'l Working Conf. Int'l Federation for Information Processing,* pp. 227-243, 2005.

[16] J. Sanchez, L. Williams, and E. Maximilien, "On the Sustained Use of a Test-Driven Development Practice at IBM," *Proc. AGILE,* pp. 5-14, 2007.

[17] L. Madeyski, "The Impact of Pair Programming and Test-Driven Development on Package Dependencies in Object-Oriented Design—An Experiment," *Proc. Seventh Int'l Conf. Product-Focused Software Process Improvement,* pp. 278-289, 2006.

[18] M. Siniaalto, "The Impact of Test-Driven Development on Design Quality," Information Technology for European Advancement, technical report, D.5.2.10 v1.0, 2006.

[19] D. Janzen, C.S. Turner, and H. Saiedian, "Empirical Software Eng. in Industry Short Courses," *Proc. 20th Conf. Software Eng. Education and Training,* pp. 89-96, 2007.

[20] M. Siniaalto and P. Abrahamsson, "A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage," *Proc. First Int'l Symp. Empirical Software Eng. and Measurement,* pp. 275-284, 2007.

[21] M. Siniaalto and P. Abrahamsson, "Does Test-Driven Development Improve the Program Code? Alarming Results from a Comparative Case Study," *Proc. Second IFIP TC 2 Central and East European Conf. Software Eng. Techniques,* pp. 143-156, 2007.

[22] D. Janzen and H. Saiedian, "Does Test-Driven Development Really Improve Software Design Quality," *IEEE Software,* vol. 25, no. 2, pp. 77-84, Mar./Apr. 2008.

[23] L. Madeyski, "The Impact of Test-First Programming on Branch Coverage and Mutation Score Indicator of Unit Tests: An Experiment," *Information and Software Technology,* vol. 52, no. 2, pp. 169-184, 2010.

[24] R. Ynchausti, "Integrating Unit Testing into a Software Development Teams Process," *Proc. Second Int'l Conf. Extreme Programming and Flexible Processes in Software Eng.,* pp. 79-83, May 2001.

[25] L. Damm, L. Lundberg, and D. Olsson, "Introducing Test Automation and Test-Driven Development: An Experience Report," *Electronic Notes in Theoretical Computer Science,* vol. 116, pp. 3-15, 2005.

[26] L. Madeyski and Ł. Szała, "The Impact of Test-Driven Development on Software Development Productivity—An Empirical Study," *Proc. 14th European Conf. Software Process Improvement,* pp. 200-211, 2007.

[27] B. George and L. Williams, "An Initial Investigation of Test Driven Development in Industry," *Proc. ACM Symp. Applied Computing,* pp. 1135-1139, 2003.

[28] B. George, "Analysis and Quantification of Test-Driven Development Approach," master's thesis, North Carolina State Univ., 2002.

[29] E. Maximilien and L. Williams, "Assessing Test-Driven Development at IBM," *Proc. 25th Int'l Conf. Software Eng.,* pp. 564-569, 2003.

[30] L. Williams, E. Maximilien, and M. Vouk, "Test-Driven Development as a Defect-Reduction Practice," *Proc. IEEE Int'l Symp. Software Reliability Eng.,* pp. 34-45, 2003.

[31] S. Edwards, "Using Software Testing to Move Students from Trial-and-Error to Reflection-in-Action," *ACM SIGCSE Bulletin,* vol. 36, no. 1, pp. 26-30, 2004.

[32] S. Edwards, "Using Test-Driven Development in the Classroom: Providing Students with Automatic, Concrete Feedback on Performance," *Proc. Int'l Conf. Education and Information Systems Technologies and Applications,* vol. 3, pp. 421-426, 2003.

[33] B. George and L. Williams, "A Structured Experiment of Test-Driven Development," *Information and Software Technology,* vol. 46, no. 5, pp. 337-342, 2004.

[34] A. Geras, "The Effectiveness of Test-Driven Development," master's thesis, Univ. of Calgary, Canada, 2004.

[35] A. Geras, M. Smith, and J. Miller, "A Prototype Empirical Evaluation of Test Driven Development," *Proc. 10th Int'l Symp. Software Metrics,* pp. 405-416, 2004.

[36] A. Hanhineva, "Test-Driven Development in Mobile Java Environment," master's thesis, Univ. of Oulu, Finland, 2004.

[37] T. Bhat and N. Nagappan, "Evaluating the Efficacy of Test-Driven Development: Industrial Case Studies," *Proc. ACM/IEEE Int'l Symp. Empirical Software Eng.,* pp. 356-363, 2006.

[38] N. Nagappan, E. Maximilien, T. Bhat, and L. Williams, "Realizing Quality Improvement through Test Driven Development: Results and Experiences of Four Industrial Teams," *Empirical Software Eng.,* vol. 13, no. 3, pp. 289-302, 2008.

[39] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. Visaggio, "Productivity of Test Driven Development: A Controlled Experiment with Professionals," *Proc. Seventh Int'l Conf. Product-Focused Software Process Improvement,* pp. 383-388, 2006.

[40] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, and C. Visaggio, "Evaluating Advantages of Test Driven Development: A Controlled Experiment with Professionals," *Proc. ACM/IEEE Int'l Symp. Empirical Software Eng.,* pp. 364-371, 2006.

[41] D. Janzen and H. Saiedian, "On the Influence of Test-Driven Development on Software Design," *Proc. 19th Conf. Software Eng. Education and Training,* pp. 141-148, Apr. 2006.

[42] D. Janzen, "An Empirical Evaluation of the Impact of Test-Driven Development on Software Quality," PhD dissertation, Univ. of Kansas, 2006.

[43] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, and H. Erdogmus, "What Do We Know about Test-Driven Development?" *IEEE Software,* vol. 27, no. 6, pp. 16-19, Nov./Dec. 2010.

[44] http://www.meta-analysis.com/, 2013.

[45] V. Kampenes, T. Dybå, J. Hannay, and D. Sjøberg, "A Systematic Review of Effect Size in Software Engineering Experiments," *Information and Software Technology,* vol. 49, nos. 11/12, pp. 1073-1086, 2007.

[46] M. Borenstein, L.V. Hedges, J.P. Higgins, and H.R. Rothstein, "A Basic Introduction to Fixed-Effect and Random-Effects Models for Meta-Analysis," *Research Synthesis Methods,* vol. 1, no. 2, pp. 97-111, 2010.

[47] A. Gupta and P. Jalote, "An Experimental Evaluation of the Effectiveness and Efficiency of the Test Driven Development," *Proc. First Int'l Symp. Empirical Software Eng. and Measurement,* pp. 285-294, 2007.

[48] L. Madeyski, "Preliminary Analysis of the Effects of Pair Programming and Test-Driven Development on the External Code Quality," *Proc. Conf. Software Eng.: Evolution and Emerging Technologies,* pp. 113-123, 2005.

[49] M. Pančur and M. Ciglarič, "Impact of Test-Driven Development on Productivity, Code, and Tests: A Controlled Experiment," *Information and Software Technology,* vol. 53, pp. 557-573, 2011.

[50] C. Desai, D. Janzen, and J. Clements, "Implications of Integrating Test-Driven Development into CS1/CS2 Curricula," *ACM SIGCSE Bulletin,* vol. 41, no. 1, pp. 148-152, 2009.
[51] 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, Springer, 2006.
[52] L. Huang and M. Holcombe, "Empirical Investigation Towards the Effectiveness of Test First Programming," *Information and Software Technology,* vol. 51, no. 1, pp. 182-194, 2009.
[53] R. Kaufmann and D. Janzen, "Implications of Test-Driven Development: A Pilot Study," *Proc. 18th Ann. ACM SIGPLAN Conf. Object-Oriented Programming, Systems, Languages, and Applications,* pp. 298-299, 2003.
[54] M.M. Müller and O. Hagner, "Experiment about Test-First Programming," *Proc. IEE Software,* vol. 149, no. 5, pp. 131-136, Oct. 2002.
[55] M. Pančur, M. Ciglarič, M. Trampuš, and T. Vidmar, "Towards Empirical Evaluation of Test-Driven Development in a University Environment," *Proc. Int'l Conf. Computer as a Tool,* pp. 83-86, 2003.
[56] S. Rahman, "Applying the TBC Method in Introductory Programming Courses," *Proc. 37th Ann. Frontiers in Education Conf.-Global Eng.: Knowledge without Borders, Opportunities without Passports,* pp. T1E-20-T1E-21, 2007.
[57] J. Vu, N. Frojd, C. Shenkel-Therolf, and D. Janzen, "Evaluating Test-Driven Development in an Industry-Sponsored Capstone Project," *Proc. Sixth Int'l Conf. Information Technology: New Generations,* pp. 229-234, 2009.
[58] S. Xu and T. Li, "Evaluation of Test-Driven Development: An Academic Case Study," *Software Eng. Research, Management and Applications,* pp. 229-238, Springer, 2009.
[59] S. Yenduri and L. Perkins, "Impact of Using Test-Driven Development: A Case Study," *Software Eng. Research and Practice,* pp. 126-129, 2006.
[60] L. Zhang, S. Akifuji, K. Kawai, and T. Morioka, "Comparison between Test Driven Development and Waterfall Development in a Small-Scale Project," *Extreme Programming and Agile Processes in Software Engineering,* pp. 211-212, Springer, 2006.
[61] T. Dogša and D. Batič, "The Effectiveness of Test-Driven Development: An Industrial Case Study," *Software Quality J.,* vol. 19, pp. 643-661, 2011.
[62] K. Lui and K. Chan, "Test Driven Development and Software Process Improvement in China," *Extreme Programming and Agile Processes in Software Eng.,* pp. 219-222, Springer, 2004.
[63] O. Slyngstad, J. Li, R. Conradi, H. Ronneberg, E. Landre, and H. Wesenberg, "The Impact of Test Driven Development on the Evolution of a Reusable Framework of Components: An Industrial Case Study," *Proc. Third Int'l Conf. Software Eng. Advances,* pp. 214-223, 2008.
[64] P. Hodgetts, "Refactoring the Development Process: Experiences with the Incremental Adoption of Agile Practices," *Proc. Agile Development Conf.,* pp. 106-113, 2004.
[65] B. Turhan, L. Layman, M. Diep, H. Erdogmus, and F. Shull, "How Effective Is Test Driven Development?" *Making Software What Really Works, and Why We Believe It,* A. Oram and G. Wilson, eds. pp. 207-219, O'Reilly Media, 2010.
[66] M. Müller and A. Höfer, "The Effect of Experience on the Test-Driven Development Process," *Empirical Software Eng.,* vol. 12, pp. 593-615, 2007.
[67] A. Höfer and M. Philipp, "An Empirical Study on the TDD Conformance of Novice and Expert Pair Programmers," *Agile Processes in Software Eng. and Extreme Programming,* pp. 33-42, Springer, 2009.
[68] L. Madeyski, "Is External Code Quality Correlated with Programming Experience or Feelgood Factor?" *Extreme Programming and Agile Processes in Software Eng.,* pp. 65-74, Springer, 2006.
[69] M. Siniaalto, "Test Driven Development: Empirical Body of Evidence," Information Technology for European Advancement, Eindhoven, The Netherlands, Agile Deliverable, vol. D.2.7, 2006.
[70] S. Kollanus, "Test-Driven Development—Still a Promising Approach?" *Proc. Seventh Int'l Conf. Quality of Information and Comm. Technology,* pp. 403-408, Oct. 2010.
[71] L. Damm and L. Lundberg, "Quality Impact of Introducing Component-Level Test Automation and Test-Driven Development," *Software Process Improvement,* P. Abrahamsson, N. Baddoo, T. Margaria, and R. Messnarz, eds. vol. 4764, pp. 187-199, Springer, 2007.

Yahya Rafique received the BSc degree from University of Toronto, Ontario, Canada, and the MSc degree from Ryerson University, Toronto, Ontario, Canada, in 2008 and 2011, respectively, both in computer science. He has experience working both as a developer and as a consultant in the professional services industry, where he is currently working. His research interests include empirical research methods in software engineering, software development methodologies and practices, software process improvement, and agile techniques.

Vojislav B. Mišić received the PhD degree in computer science from the University of Belgrade, Serbia, in 1993. He is a professor of computer science at Ryerson University in Toronto, Ontario, Canada. His research interests include performance evaluation of wireless networks and systems, and software engineering; he has published more than 60 journal papers and more than 130 conference papers in these areas, in addition to six books and nearly 20 book chapters. He serves on the editorial boards of *IEEE Transactions on Parallel and Distributed Systems*, *Ad Hoc Networks*, *Peer-to-Peer Networks and Applications*, and the *International Journal on Parallel, Emergent and Distributed Systems*. He is a senior member of the IEEE, and a member of the IEEE Computer Society, ACM, and AIS.