Information and Software Technology 56 (2014) 375–394

Contents lists available at ScienceDirect

Information and Software Technology

journal homepage: www.elsevier.com/locate/infsof

Considering rigor and relevance when evaluating test driven development: A systematic review

Hussan Munir[a], Misagh Moayyed[b], Kai Petersen[c,*]

[a] Lund University, Department of Computer Science, P.O. Box 118, SE-221 00 Lund, Sweden
[b] Unicon Inc, AZ, USA
[c] School of Computing, Blekinge Institute of Technology, Karlskrona, Sweden

ARTICLE INFO

Article history:
Received 15 July 2013
Received in revised form 4 January 2014
Accepted 5 January 2014
Available online 13 January 2014

Keywords:
Test-driven development (TDD)
Test-last development (TLD)
Internal code quality
External code quality
Productivity

ABSTRACT

Context: Test driven development (TDD) has been extensively researched and compared to traditional approaches (test last development, TLD). Existing literature reviews show varying results for TDD.
Objective: This study investigates how the conclusions of existing literature reviews change when taking two study quality dimension into account, namely rigor and relevance.
Method: In this study a systematic literature review has been conducted and the results of the identified primary studies have been analyzed with respect to rigor and relevance scores using the assessment rubric proposed by Ivarsson and Gorschek 2011. Rigor and relevance are rated on a scale, which is explained in this paper. Four categories of studies were defined based on high/low rigor and relevance.
Results: We found that studies in the four categories come to different conclusions. In particular, studies with a high rigor and relevance scores show clear results for improvement in external quality, which seem to come with a loss of productivity. At the same time high rigor and relevance studies only investigate a small set of variables. Other categories contain many studies showing no difference, hence biasing the results negatively for the overall set of primary studies. Given the classification differences to previous literature reviews could be highlighted.
Conclusion: Strong indications are obtained that external quality is positively influenced, which has to be further substantiated by industry experiments and longitudinal case studies. Future studies in the high rigor and relevance category would contribute largely by focusing on a wider set of outcome variables (e.g. internal code quality). We also conclude that considering rigor and relevance in TDD evaluation is important given the differences in results between categories and in comparison to previous reviews.

Contents

* Corresponding author. Tel.: +0046 (0) 455 385877.
E-mail addresses: hussan.munir@cs.lth.se (H. Munir), mmoayyed@unicon.net (M. Moayyed), kai.petersen@bth.se (K. Petersen).

0950-5849/$ - see front matter © 2014 Elsevier B.V. All rights reserved.
http://dx.doi.org/10.1016/j.infsof.2014.01.002

1. Introduction

TDD is considered as a key practice of extreme programming (XP) [1] and incorporates refactoring [2]. Later on TDD as a practice was evaluated in isolation [3,4] and received much attention in empirical studies. The publications by [1,2] have been seminal to inform how TDD is conducted. The single most important rule in TDD is: *"if you can not write a test for what you are about to code then you should not even be thinking about coding"* [5]. That is, in contrast to test last development (TLD) the test cases are written first prior to writing the code of the software. The process of TDD consists of the following steps [6]: First, a feature or a user requirement is selected. Thereafter, a test is written that fulfills a small task or piece of the feature or user requirement (e.g. one method) and fails. Next, the production code is written that implements the functionality to be tested. Having implemented the functionality all tests are run. If any test fails the production code is corrected and the entire test suite re-run. If the tests pass the production code and tests are re-factored to make them as simple as possible. The difference between TDD and TLD is illustrated in Fig. 1.

In previous literature reviews [7–11] TDD literature has been aggregated. The findings from the literature reviews are contradictory. Four reviews [7,11,9,10] focused on evaluating the outcome of using TDD with respect to different outcome variables. Emphasis have been given to quality, namely external quality (also referred to as delivered quality [9], internal code quality, and productivity). One more review has been conducted on TDD, but with different focus. Causevic et al. [8] investigated inhibitors for adopting TDD in industry.

Previous literature reviews on TDD have not utilized a wide range of quality criteria when interpreting the results of TDD studies. The criteria were used in study selection, however, the degree of fulfilling different criteria has not been considered in the analysis and interpretation of the results. Ivarsson and Gorschek [12] emphasized that it is important to consider two dimensions, namely rigor and relevance. They provided a rubric that allows scoring of both dimensions resulting in studies receiving a value between 0 and 3 for rigor and 0 and 4 for relevance. This enables researchers to understand the conclusions of studies in relation to different and fine grained levels of quality. Rigor and relevance are defined as follows (cf. [12]):

- Overall rigor is concerned with adhering to good practices when applying a research method and reporting on them. Rigor defines the extent to which context, study design, and validity are described and reflected upon.
- Relevance relates to the practical impact and realism of the research setup. Relevance considers whether the results apply to an industrial context and are relevant for it. In particular subjects, context, research method, and scale are assessed.

The extent of fulfilling the criteria is rated on an ordinal scale. Given that studies achieve different levels of rigor and relevance there is a need to re-evaluate conclusions from previous literature reviews. In particular, the distinction between rigor and relevance adds an important angle for interpretation. For example, a study might be highly relevant (e.g. lessons learned from a large industrial development project), but at the same time has very low rigor (e.g. missing validity evaluation, and lack of description of data collection approaches).

In response to the above mentioned needs we conducted a systematic review on literature comparing TDD with TLD taking rigor and relevance into consideration. In particular, this study makes the following contributions:

- Contribution 1: Identification of outcome variables based on which TDD has been evaluated. Characterizing the variables and how they are quantified (measured) is an essential first step in order to determine the evaluation criteria for TDD.

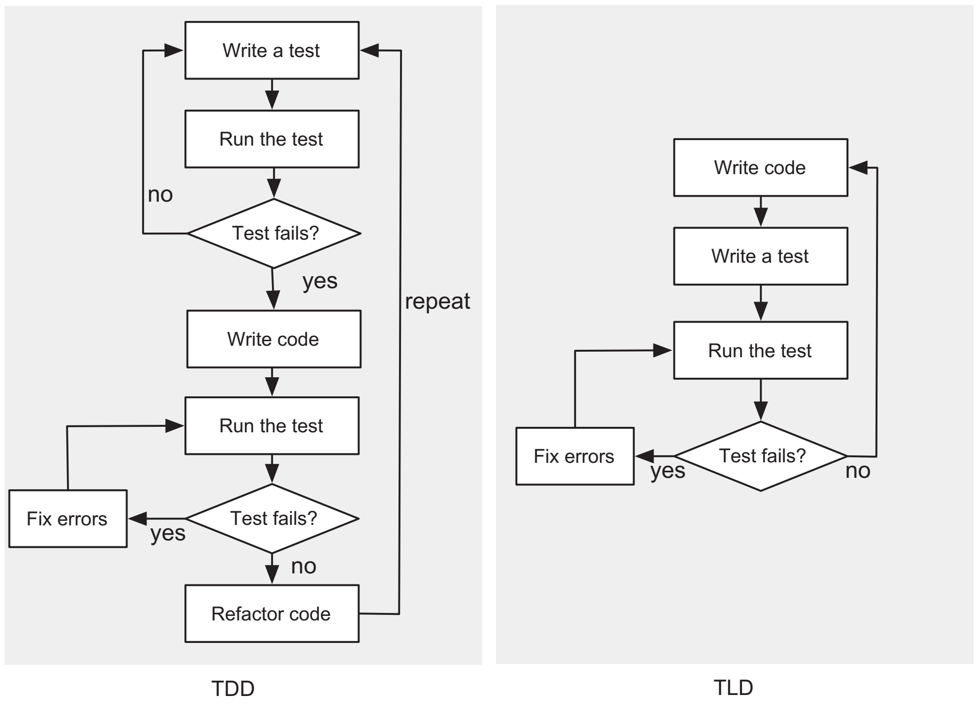

Fig. 1. TDD vs. TLD (based on [87]).

- Contribution 2: Identification of outcomes for individual primary studies with respect to the outcome variables and their interpretation in the light of rigor and relevance. Studies are categorized in four groups: high rigor and high relevance, high rigor and low relevance, low rigor and high relevance, and low rigor and low relevance. The outcomes in each category are compared.
- Contribution 3: The outcome of this literature review is compared with the outcome of previous reviews. This provides new insights if the conclusions change and also adds further evidence for the usefulness of using the assessment instrument proposed by [12].

Overall, this assessment has practical relevance as companies should base their decisions on those studies with high rigor and relevance [12], in particular in case of conflicting results, as presented in previous literature reviews on TDD.

The remainder of the paper is structured as follows: Section 2 presents related work. Section 3 presents the research method (review protocol). Section 4 provides the results, followed by a Discussion (Section 5) and Conclusion (Section 6).

2. Related work

We identified 8 works on test-driven development with the reviews being the main or a major part of the contribution (cf. [6,8,13,7,10,9,14,15]). One review focuses on limiting factors that hinder the adoption of TDD [8], while we are focusing on the actual outcome achieved by using TDD in comparison to TLD. Hence, we discuss studies [13,7,10,9,14,15] in detail as those are directly related and most relevant to our systematic review, using the questions suggested in [16,17]. The results of the systematic review by Shull et al. [9] are based on [18].

What are the reviews' objective? Kollanus conducted a systematic review reported in two papers [13,7]. He analyzed empirical studies with respect to quality and productivity. While doing so, he distinguished different study types (in particular case study, experiment, and others).

Shull et al. [9] assessed internal code quality, external code quality and productivity. They assessed the results considering the relevance dimension by classifying a sub-set of studies as being of high quality. This has been done through defining criteria similar to what Ivarsson and Gorschek [12] consider as the relevance dimension. Shull et al. highlighted studies when subjects are on graduate or professional level, followed the TDD text-book process, and the size of the development task was realistic.

Sfetsos and Stamelos [10] reported on the impact of agile practices on quality, including TDD, pair programming, and other agile practices.

Rafique and Misic [14] conducted a statistical meta-analysis to determine the effects of TDD on external quality and productivity.

Jeffries and Melnik [15] aimed at summarizing the outcomes of TDD studies in academia and industry in relation to quality and productivity impacts.

We complement the existing studies in the following ways: We extend all studies by eliciting a wider set of outcome variables. Furthermore, we complement the study by Shull et al. [9] by adding the dimension of scientific rigor, and extend the relevance dimension to consider whether the evaluation has been done on a real industrial system, the research methodology is industry focused, and that the research context in which the study is conducted matches real usage (industry). The rigor and relevance dimensions provide a more fine-grained scale for rigor and relevance. The scale is used to compare sets of studies with different levels of quality with each other. Further details on the quality assessment can be found in Section 3.7 and are based on [12].

What sources were selected to identify the primary studies? Were there any restrictions? Was the search done on appropriate databases and were other potentially important sources explored? What were the restrictions and is the literature search likely to have covered all relevant studies?

Kollanus [13,7] used IEEE Explore, ACM, and Springer. The time period has not been specified. The search terms used were "TDD" and "test-driven development".

Shull et al. [9] have not specified the data bases and the search string. The time period covered papers from 1999.

Sfetsos and Stamelos [10] searched in IEEE Explore, ACM, Kluwer, SpringerLink, ScienceDirect, ISI Web of Science, CiteseerX, and Wiley. The search terms aimed at identifying agile practices, pair programming, and TDD studies. Furthermore, the term "*empirical*" was used to restrict the search to studies using empirical research methods. The time period covered was up till and including 2009.

Rafique and Misic [14] searched ACM Digital Library, IEEE Xplore, SpringerLink, ISI Web of Science, and Scopus. Their search focused on identifying all studies related to TDD, using the terms "Test Driven Development", "Test-Driven Development" and TDD.

Jeffries and Melnik [15] have not provided information on sources and search.

We complemented our database search (see Section 3.3) with snowball sampling (cf. [19,20]). Snowball sampling refers to checking the reference lists of included papers for additional references (also referred to as backward snowball sampling). With regard to the use of data-bases we included index databases (ISI Web of Science and Inspec/Compendex) as well as publisher databases (ACM, IEEE, Science Direct). A similar strategy was followed by Sfetsos and Stamelos [10] and Rafique and Misic [14], while Kollanus [13,7] only focused on publisher databases.

Are the inclusion and exclusion criteria defined and appropriately applied? Kollanus [13,7] focused the inclusion and exclusion on whether the study was about TDD.

In [9] no inclusion and exclusion criteria have been mentioned, however, quality checks have been referred to.

Sfetsos and Stamelos [10] focused the selection on agile practices and that they have been evaluated with respect to their impact on quality.

Rafique and Misic [14] defined criteria to identify studies that allowed to conduct a statistical meta-analysis, which required a control group and a quantitative component. Internal quality attributes were not considered as they only indirectly influence external quality.

What quality criteria were used to assess the quality/validity of the included studies?

Kollanus [13,7] did not define quality criteria or conducted an assessment.

Shull et al. [9] conducted a quality assessment, however, the detailed criteria were not defined. The relevance dimension (practitioners, process adherence, and size of the task) can be considered a quality instrument and has been used for the interpretation of the results.

Sfetsos and Stamelos [10] assessed how well the research methods have been used (e.g. whether the study was empirical, the research design has been described, a control group has been used, etc.) and assessed the credibility of the results (clarity of findings, effect of the researcher on the interpretation). The scoring of each criterion was binary. A quality threshold for study selection has been mentioned, but no further details have been provided. The results of the quality assessment have not been used in the interpretation of the results.

Rafique and Misic [14] defined quality criteria in relation to rigor and extraction context and output information.

No inclusion and exclusion criteria are defined in Jeffries and Melnik [15].

In our study we complement the existing studies by using the clearly distinct rigor and relevance dimensions to evaluate the quality of the primary studies. The quality assessment plays an essential role in the interpretation of the results, outcomes of different quality levels being compared with each other. This allows to determine whether combining sets of studies of different quality levels would bias the results if they came to very different conclusions.

How were the data extracted from the primary studies?

Kollanus [13,7] used a data extraction form containing research question, empirical method, brief description of study, main results, evidence on TDD, critical comments, and how TDD was introduced.

In [9] the data extraction form has not been presented.

Sfetsos and Stamelos [10] extracted title, abstract, type of empirical study (experiment, case study, survey, mixed, etc.), research environment, population, agile practice in use, research question, main results, evidence with respect to quality, and study conclusions.

Rafique and Misic [14] elicited context (author, number of participants, academia or industry), rigor (process of testing done by control group, other agile practices, experience, task size, duration, process conformance information, training), and outputs (p-values, mean and standard deviation, % of improvement).

Jeffries and Melnik [15] do not define the process for extracting the data. In their extraction they focused on research type, development time, legacy project, organization, software type, size, number of participants, programming language, and outcomes (productivity and quality effect).

The data extraction form used in this study is presented in Section 3.6.

Were the basic data/studies adequately described? Sfetsos and Stamelos [10], Jeffries and Melnik [15] as well as Rafique and Misic [14] provided details on individual studies, summarizing individual studies in a table.

We provided the details of each individual studies for each rigor and relevance category (see Appendices B,C,D,E).

How were the data synthesised? How were differences between studies investigated? How were the data combined? Was it reasonable to combine the studies? Kollanus [13,7] as well as Shull et al. [9] used vote counting. Studies were counted with respect to categories related to the effect (positive, negative, or indifferent) for the evaluated outcome variables (quality and productivity).

Sfetsos and Stamelos [10] provided narrative summaries and tabulated the results of individual studies.

Rafique and Misic [14] used statistical meta-analysis focusing on percentage of improvement as an effect size measure.

Jeffries and Melnik [15] provide detailed information about study context and outcome through tabulation.

In this study we used vote counting, which aids in the comparability between this study and the previously conducted studies. The vote counting in this study is done for studies on different quality levels based on their rigor and relevance score.

How sensitive are the results to the way the review has been done? Varying levels of detail have been provided on the mode of collaboration between the authors involved in previous literature studies. Studies [13,7] are an effort of an individual author, which poses a validity threat as no discussion or review between authors is possible. The study by Shull et al. [9] was a collaborative effort. Sfetsos and Stamelos [10] explicitly refer to discussions and disagreement resolution in their review process. Rafique and Misic [14] clearly defined the analysis approaches, which were of statistical nature, focusing on quantitative outcomes, which can be objectively extracted. The mode of collaboration between authors has not been described. In Jeffries and Melnik [15] the process for conducting the review has not been defined.

In this study, we used multiple reviewers for the study selection and quality assessment steps. In the study selection, the first and second author individually selected the studies and disagreements and uncertainties were later discussed. For the quality assessment, the third reviewer reviewed the data extraction and ratings of rigor and relevance. Furthermore, selection criteria, forms, and quality criteria were discussed and reviewed among the authors to assure a common understanding.

Table 1
Overview of previous literature reviews.

Study	Purpose	Studies identified	Major outcomes
Shull et al. [9]	Examine TDD's effect on outcome variables	33 studies (13 controlled experiments, 18 industrial use)	Contradictory results for internal code quality and productivity, primarily positive results for external code quality
Kollanus [11,7]	Presents a number of possible issues and challenges with TDD that are referred in the literature	40 studies	Contradictory results for all variables (internal code quality, external code quality, productivity); study concluded that there is some evidence on better code quality but decreased productivity with TDD. However, primary studies are not comparable due to heterogeneous nature of variables and overall the empirical evidence on TDD is weak
Sfetsos and Stamelos [10]	This study attempts to present and evaluate the empirical findings regarding quality in agile practices	46 studies (24 experiments, 17 case studies, 5 mixed)	The main focus of study was to investigate the empirical evidence on quality in agile practices. The findings suggest that agile practices can improve the quality if they are implemented correctly. Pair programming was found to be a successful practice in combination with TDD/TFD to improve the code quality. However, the productivity effects of TDD were contradictory, and the results varied for different context
Rafique and Misic [14]	Statistical meta-analysis with respect to external quality and productivity	26 studies	Small improvements in quality and inconclusive results for productivity. Industrial experiments lead to more significant improvements in quality
Jeffries and Melnik [15]	Summarize and tabulate TDD studies with respect to productivity/effort and quality	18 studies	For quality positive results dominate, while for productivity negative results dominate

Table 1 provides an overview of existing literature reviews on TDD, including their main findings.

In summary, our systematic review complements the existing reviews by identifying a wider set of variables and how they have been measured and using a recently introduced instrument to aggregate evidence with respect to rigor and relevance [12]. New findings have been obtained that provide new insights in comparison to the previous studies. In all stages of the review a systematic approach has been followed to increase the reliability of the results as suggested in [21,22]. Kitchenham et al. further underline the importance of following a systematic process in [23] based on a case study.

3. Research method

The research method used is systematic literature review following the guidelines in [21]. Systematic reviews have the benefit of documenting a repeatable research process, and focus on systematic evidence aggregation. Furthermore, an important aspect is to consider the strength of evidence, which in this case is done by using the rubric provided in Ivarsson and Gorschek [12] to assess rigor and relevance. In the following sections the review protocol of this study is documented. This systematic review is solely based on the information documented in the articles.

3.1. Research questions

We formulated 3 research questions and shortly state how they help to achieve the contributions mentioned earlier and also argue for their relevance.

RQ1: Which variables have been considered to capture benefits and limitations of TDD? The answer to this questions helps to identify the outcome variables that have been investigated in the previous studies along with their measurement units.
RQ2: How does TDD perform with respect to the outcome variables? The answer to this question provides an insight on the performance of TDD with respect to the outcome variables identified in RQ1.
RQ3: How strong is the evidence with respect to rigor and relevance to support the outcomes with respect to different variables? The answer to this question allows us to assess the strengths of evidence based on rigor and relevance with respect to the outcome variables (RQ2). As a hypothetical example, if studies with a positive outcome for TDD productivity have high rigor and relevance, while negative ones do not, then the studies with the positive outcome weigh higher [12]. That is, high rigor and high relevance studies should be given a higher weight when giving recommendations to practitioners.

3.2. Identification of primary studies

In the review process, the following steps were undertaken to arrive at the final set of primary studies (see Fig. 2).

1. Extract studies from the databases after applying the search string: A total of 2007 papers has been identified based on the search (see Section 3.3).
2. Remove duplicate studies: 463 duplicates have been removed through identification by the reference management system (EndNote).
3. Select relevant studies based on title, abstract and keywords: After applying inclusion and exclusion criteria on title, abstract, and keywords, 1435 additional articles were excluded, which resulted in 109 papers for full-text reading. The inclusion and exclusion criteria are presented in Section 3.4.
4. Select studies based on detailed review/screening: The detailed screening led to the removal of 66 additional papers. Furthermore, 5 papers had to be excluded due to that the full text was not retrievable. The criteria for the detailed screening are presented in Section 3.5.
5. Apply backward snowball sampling: Through snowball sampling [20,19] only three additional studies were identified. However, given that previous studies found that snowball sampling is capable of finding studies that are not captured during database searches [20,19] it is an important practice to be used in study identification.

The final set of papers included in the study is 41 papers.

For these papers data has been extracted (see Section 3.6) and has been analyzed and assessed for rigor and relevance (cf. [12]).

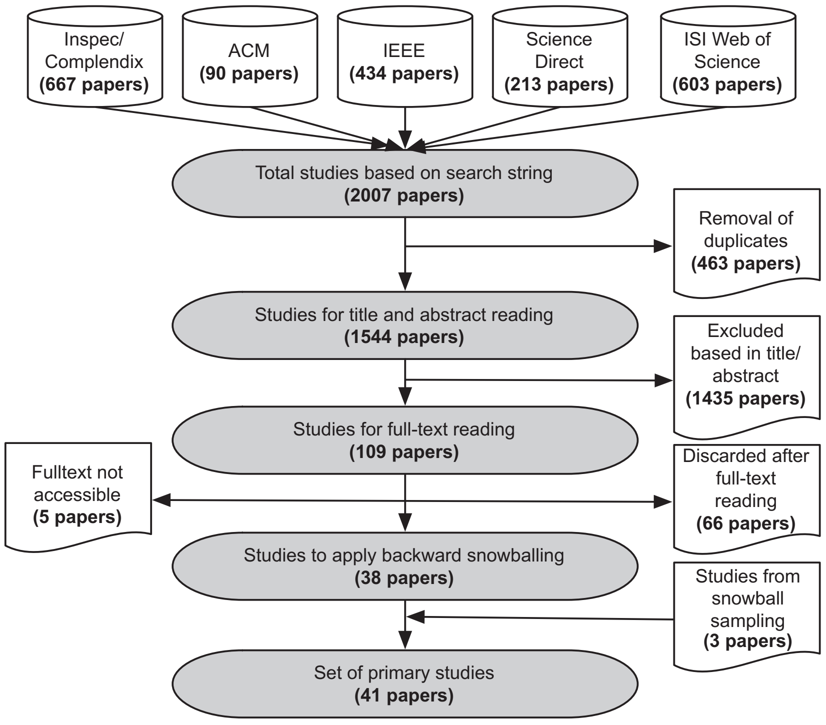
Inspec/ Complendix (667 papers)
ACM (90 papers)
IEEE (434 papers)
Science Direct (213 papers)
ISI Web of Science (603 papers)
Total studies based on search string (2007 papers)
Removal of duplicates (463 papers)
Studies for title and abstract reading (1544 papers)
Excluded based in title/ abstract (1435 papers)
Studies for full-text reading (109 papers)
Fulltext not accessible (5 papers)
Discarded after full-text reading (66 papers)
Studies to apply backward snowballing (38 papers)
Studies from snowball sampling (3 papers)
Set of primary studies (41 papers)

Fig. 2. Paper review process.

3.3. Search string strategy

In order to develop the search string keywords were extracted from the 20 known primary studies on TDD and organized them into intervention (S1, including terms referring to TDD and test last development), research method (S2), and outcomes (S3):

- *S1:String related to Test-Driven Development and Test-Last Development.* Test Driven development **OR** Agile Driven **OR** Test Driven design **OR** Test First **OR** TDD **OR** Unit Test.
- *S2: String related to systematic literature review, case study, interview, survey and experiment design/type.* Experiment* **OR** investigation **OR** empirical **OR** Systematic literature review **OR** revision **OR** interview **OR** survey **OR** case study.
- *S3: String related to outcomes.* Code quality **OR** productivity **OR** Limiting factor **OR** benefit **OR** disadvantage **OR** advantage.

We also piloted the search including the term *"experience report"* in S2 (research method) in Scopus without success in obtaining a wide range of results that would affect the overall conclusions. Hence, experience reports have not been included as a search term.

The overall string was combined as: (Test Driven development **OR** Agile Driven **OR** Test Driven design **OR** Test First **OR** TDD **OR** Unit Test) **AND** (Experiment* **OR** investigation **OR** empirical **OR** Systematic literature review **OR** revision **OR** interview **OR** survey **OR** case study) **AND** (Code quality **OR** productivity **OR** Limiting factor **OR** benefit **OR** disadvantage **OR** advantage), resulting in a relatively high number of initial papers. This was tolerated with the aim of being inclusive. Using the search databases, 17 of the 20 known TDD studies (studies known as being relevant to the authors prior to conducting the study) were identified. As pointed out by Wohlin and Prikladnicki [24] it is likely relevant papers are missed using only one search strategy. However, they emphasize that it is important to get a good sample of the overall population of relevant papers. To achieve this it is suggested to complement different strategies. In this study, given that database searches often miss relevant papers, we complemented the search with snowball sampling [20]. Furthermore, in order to evaluate the quality of our search and selection process, we compared the sets of selected studies with those of existing literature reviews to determine the quality of our overall result.

The following research databases were searched:

- ACM Digital Library.
- IEEE Xplore.
- Inspec and Compendix (Engineering Village), Meta-database searching multiple publishers.
- Science Direct (Elsevier).
- ISI Web of Science, Meta-database searching multiple publishers.

The study did not include the research papers from software engineering related databases such Kluwer Online, SpringerLink, Scopus and Wiley Interscience since the previously conducted studies have revealed that the research paper extracted from these databases are also returned by either Engineering Village or ISI Web of science [25,26].

3.4. Inclusion/exclusion criteria for title/abstract reading

In order to pick the final set of primary studies from the complete set of extracted papers from selected databases we considered following inclusion/exclusion criteria:

- Studies present results on TDD.
- Studies are peer-reviewed (journals, conferences, and workshops).
- Studies are accessible in full-text.

- Studies report the outcome variables for TDD along with their measuring criteria (i.e. if there is an improvement, no difference, or a decline).
- Studies are empirical.

Articles are excluded if they fulfill one of the following exclusion criteria:

- Studies are not related to TDD.
- Studies are non-empirical, i.e. they do not provide qualitative/quantitative evidence, including experience reports.
- Studies are not presented in English.
- Studies are not accessible to the authors.
- Studies are not peer reviewed.
- Studies are duplicates.

The authors rated the papers independently as include, exclude, or uncertain. In case of disagreements or uncertainty the authors discussed the paper, if they are still uncertain during the discussion the paper was included in the next step. The purpose of using an inclusive strategy is to reduce the risk of excluding relevant papers, as recommended by Ali and Petersen [22].

3.5. Exclusion for detailed screening

Given that the abstract does not always reveal whether there is data available in the paper that supports a benefit or limitation with respect to outcomes, further papers had to be excluded (see Step 5 in Fig. 2). The following questions aid in making the decision of whether to exclude an article at this stage:

- Does the study state a research question/goal or research hypothesis?
- Does the study report data related to its outcome variables (benefits or limitations)?
- Does the study report the measuring criteria for the outcome variables, so that changes in performance can be identified?

Kappa analysis [27] was performed to particularly measure the agreement level between researchers on a test-set of 20 articles included in the previous step. In particular, we rated the research method applied in the full-text article as case study, survey, experiment, interview, and literature review. The exit criterion for the primary study selection process is to gain substantial agreement levels between the researchers as a result of Kappa analysis. The measured Kappa agreement level based on the initial rating of the papers is calculated to be 0.598, which shows a moderate strength of agreement (cf. [27]). Our aim was originally to reach a substantial level of agreement in categorization of the studies.

Hence, we discussed all disagreements as suggested in [22], and subsequently resolved disagreements and uncertainties after discussion.

In order to check the reliability of the selected final set of primary studies we cross-checked the final selection of our studies with the reference lists of [7,10], which were the reviews relevant for this work that provided a reference list.

Table 2 shows the studies we missed from the literature reviews, and vice versa. Missing a study could either mean that it was not found in the search or not included (given that excluded papers were not documented). Studies identified by us, but missed by others, are from before the publication date, so that they could have potentially been found. Studies identified by others, but were missed by us, are counted as missed when fitting our inclusion criteria. The study missed from Sfetsos and Stamelos is also in the set of Kollanus. Rafique and Misic had strict inclusion criteria (availability of quantitative data for statistical meta-analysis), resulting in excluded studies. We assessed the missing studies identified post-finalization of our literature review, and determine their impact on the conclusions (see Section 5.4).

It is interesting that, despite three independent researchers conducting a search and study selection with similar intentions, studies are still missing in all studies. Similar findings have been observed where [28] investigated two systematic mapping studies on the same topic (product line testing) that have been conducted independently by researchers from different institutions. Similar findings have also been obtained by [23].

Table 2
Studies missed in literature reviews.

Study	Description	No. of studies
Kollanus [11]	Studies we missed	4
	Studies Kollanus missed	9
Sfetsos and Stamelos [10]	Studies we missed	1
	Studies Sfetsos and Stamelos missed	21
Rafique and Misic [14]	Studies we missed	4
	Studies Rafique and Misic missed	28
Jeffries and Melnik [15]	Studies we missed	2
	Studies Jeffries and Melnik missed	31

3.6. Data extraction strategy and analysis

We prepared the list of attributes that needed to be extracted from studies in order to answer proposed research questions for this study. To make the data extraction consistent from all studies an excel sheet was created (see Table 3).

Based on the extraction the results are tabulated to capture research method, context, data sources, reported benefits, and reported limitations. Thereafter, the studies are divided into different categories depending on their relevance and rigor scores. A studies have a high rigor and relevance score, B studies have either high rigor and low relevance, or vice versa. C studies have low rigor and low relevance Vote counting per each outcome variable (e.g. productivity, external code quality, etc.) is then used to determine either positive, indifferent, or negative results for each variable. Given that the studies are classified based on their relevance and rigor scores the analysis will reveal outcomes in relation to strength of evidence.

3.7. Evaluation of study quality through rigor and relevance

Quality assessment has been conducted according to the rubrics defined in [12] with rigor and relevance being the two main criteria for assessment.

Table 3
Data extraction strategy.

Category	Properties
General information	Title, author(s), publication year, abstract
Research methods	Systematic review, case study, survey, experiment, mixed
Study outcome variables	Internal code quality, external code quality, productivity, benefits of TDD
Statistical data collection (If applicable)	*P* values, standard deviation, mean value, *T*-test value
Number and type of subjects	Students, professionals, mixed
Results	Limiting factors of TDD, significant or insignificant difference

Rigor: Rigor can be defined as *"the research methodology is carried out in accordance with corresponding best practices"* (cf. [12]). These can be found in guidelines for different research methods (e.g. [29] for experimentation and [30] for case studies). According to [12] rigor has two dimensions, namely following the best practices and complete reporting of the study. In this literature review we assess the papers based on what has been reported. Through aggregating of study presentation aspects from previous literature Gorschek and Ivarsson [12] defined rigor as the degree to which study context (C), design (D), and validity threats (V) were described. Each aspect is rated on a scale for description, namely weak, medium, and strong description.

3.7.1. Context (C)

1. **Strong description:** The context is sufficiently described to be comparable to other settings [12,31]. In particular, we emphasized subject type (graduate, undergraduate, professionals), development experience, development methodology, duration of the observation. If all these factors are presented C evaluates to 1.
2. **Medium description:** If any of the factors is missing C evaluates to 0.5.
3. **Weak description:** If neither of the aforementioned factors is found in the study then C evaluates to 0.

3.7.2. Design (D)

1. **Strong description:** The research design is transparent and detailed enough for the reader to understand the design [12], i.e. outcome variables, measurement criteria, number of subjects, treatments, and sampling is defined, D evaluates to 1.
2. **Medium description:** If an important aspect related to design and data collection is missing (see above) then D evaluates to 0.5.
3. **Weak description:** If no design description is provided D evaluates to 0.

3.7.3. Validity threats (V)

1. **Strong description:** If different types of validity (internal, external, conclusion, and construct validity) are evaluated and reflected upon V evaluates to 1.
2. **Medium description:** If only a subset of the relevant threat categories is discussed V evaluates to 0.5.
3. **Weak description:** In case no validity discussion is provided V evaluates to 0.

Relevance: Relevance is related to the actual impact on industry [12]. This relates to multiple aspects, namely the relevance of the topic studied [32]; the ability to apply a solution in an industrial setting with success (live situation) [33]; the use research methods that facilitate realism and hence are more interesting for practitioners [34]; the presence of a realistic situation with respect to users, scale, and context [12]. The approach of [12] considers the following aspects for rigor: users/subjects (U), scale (S), research methodology (RM), and context (C).

3.7.4. Users/subjects (U)

1. **Contribute to relevance:** The subjects are representative of the actual users, i.e. U is evaluates to 1 for industry professionals.
2. **Partially contribute to relevance:** The subjects are partially representative, i.e. they are M.Sc. or graduate studies, U evaluates to 0.5.
3. **Does not contribute to relevance:** The subjects are not representative, i.e. they are bachelor/undergrad students or the information is missing, U evaluates to 0.

3.7.5. Scale (S)

1. **Contribute to relevance:** The used application scales to industry complexity and is representative for a real system, S evaluates to 1 if an industry application/real system is used.
2. **Does not contribute to relevance:** The application is down-scaled or a toy example, and hence not representative, S evaluates to 0.

3.7.6. Research Methodology (RM)

1. **Contribute to relevance:** The research methodology is suitable to investigate real world contexts and situations with relevance for practitioners (action reserach, case study, industry interviews, experiment investigating a real situation, and surveys/interviews), RM evaluates to 1.
2. **Does not contribute to relevance:** Lab experiment (human subjects/software) or not reported evaluates RM to 0.

3.7.7. Context (C)

1. **Contribute to relevance:** The study is investigated in a setting that matches real industrial usage (industrial setting), C evaluates to 1.
2. **Does not contribute to relevance:** An artificial setting (e.g. lab) is created or others that do not represent a context matching real world situations, or not reported evaluates C to 0.

The solution for evaluating rigor and relevance resembles the use of a rubric based evaluation in education [12]. It was found that rubrics increase the reliability of assessments in terms of inter-rater agreement [35–37]. In order to further increase the confidence given by using a rubric the third reviewer verified the rating and data extraction through review by identifying the information extracted in the original paper.

The clear distinction between the rigor and relevance dimension, as well as the ability to rate each dimensions on a scale, motivated us to choose this quality assessment framework. The rubric is also well defined so that, if many studies use the same scoring system, comparability between different areas of study are possible. However, the rubric also has limitations as discussed in Section 5.5.

4. Results

A total of 41 studies have been found that investigated the positive or negative impact of TDD compared to TLD. All these studies were published from the year 2000 to 2011. The inspected publications were classified first according to the applied research methods. Experiments (25, 61%) and case studies (13, 32%) constitute a clear majority of studies, followed by surveys (3, 7%).

Table 4 provides a mapping of research methodologies to the primary studies included for analysis.

Table 4
Mapping of research methods to studies.

Method	Studies	No. of studies
Experiments	[38,5,6,39–56,3,13,57,58]	25
Case studies	[59–71]	13
Surveys	[72–74]	3

4.1. RQ1: Variables to assess benefits and limitations

An objective of the study was to identify the themes and patterns in the studies reported on TDD. As can be seen in Fig. 3, 8 categories were found in the studies that investigated the impact of TDD over TLD.

The categories are as follows:

1. Productivity.
2. External quality.
3. Developer opinion.
4. Internal code quality.
5. Effort/time.
6. Conformance.
7. Size.
8. Robustness.

Productivity, effort/time, and size: Productivity (in general defined as the relation between output and input [75]) in the majority of studies indicates a measure of team speed that is calculated by a variety of elements including lines of code per month, number of written unit tests and a measured level of development effort calculated in units of time. It should be noted that productivity is generally discussed in very broad terms and each study independently defines a unique measure that is sometimes inconsistent with the others. Effort and size were initially considered as sub categories of productivity since some of the studies in productivity have calculated the size and development effort. However, studies were found that investigated the effort and size as separate variables but consistency was not found in terms of measuring criteria.

Quality (internal and external): External code quality refers to the delivered quality [9], indicated primarily by defects and passing test cases. Internal code quality is a metric that broadly evaluates internal program design, code and test coverage, flexibility of system architecture decisions and various levels of code coupling and cohesion that directly affect productivity and code maintenance. In TDD context, design is encompassed under the umbrella of internal code quality in general and attempts to structure the system in an evolutionary fashion that allocates less up-front planning time and focus more on frequent code re-factoring to prevent architecture erosion. Internal code quality deals with the quality of the code in terms of code complexity (Cyclomatic Complexity), test coverage, coupling between objects, lack of cohesion method, information flow and nested block depth etc.

Developer opinion: In order to investigate the programmers' opinion, studies have used Likert scales [76] from 1 to 10, exam scores, questionnaires and surveys based on choices such as fewer defects, correctness, simpler, best choice and through testing etc. However, the majority lacks consistency across studies in terms of choices.

Conformance: Conformance is investigated in studies by defining the rules for TDD. First, a change in the code is only allowed in methods which were previously called by a test that ended up as a failure, thus causing overall unit test failure. Second, a new method can be introduced if it is later called by a test that fails thus causes the whole unit test to fail. Third, a refactoring can only change the structure of code rather than behavior of the code.

Robustness: Studies also investigated the use of TDD in resolving robustness problems, such as presenting unexpected behavior in the presence of invalid inputs. The distinct patterns or themes have been identified based on experiments and mixed research methods. The measuring criteria are associated with experimental studies since the case studies did not report statistical metrics used for the measurement, given that they were qualitative in nature.

4.2. RQ2/3: Benefits and limitations in relation to rigor and relevance

Each study has been scored according to rigor and relevance criteria, as described in Section 3.7. Tables A.13 and A.14 (see Appendix A) provide an overview of the scores achieved by individual studies with respect to rigor and relevance. Table A.13 is composed of columns representing the elements of rigor, namely description of context (C), study design description (S), and validity threats (V).

Table A.14 illustrates the components of relevance, namely users/subjects (U), context (C), research method (RM), and scale (S).

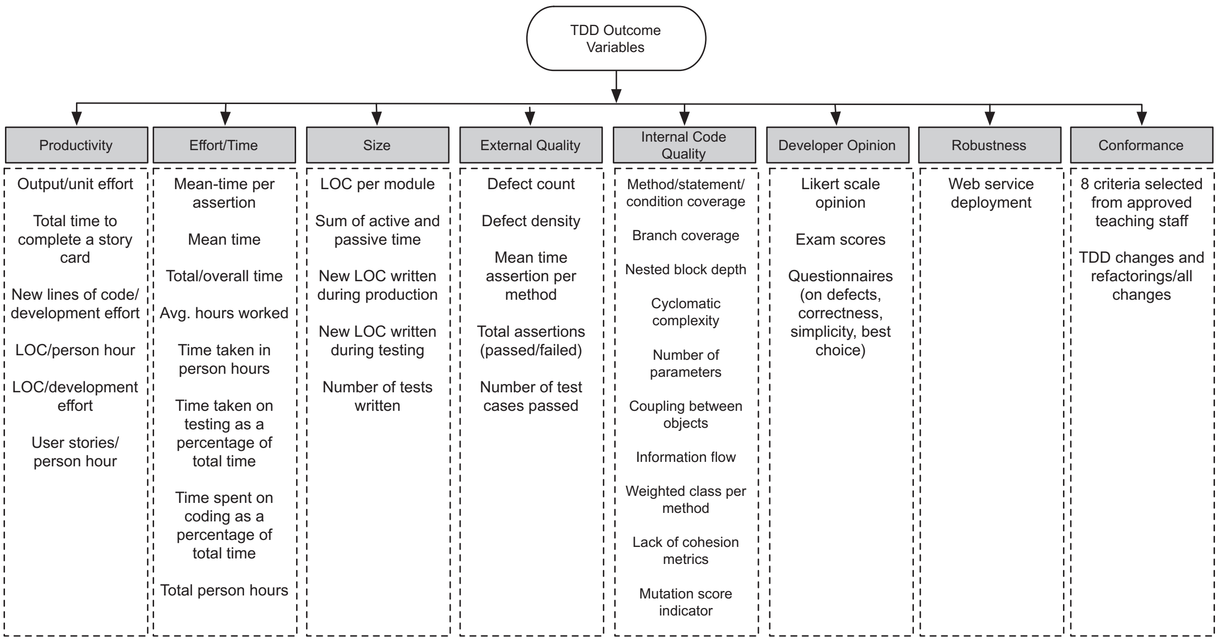
TDD Outcome Variables
Productivity
Output/unit effort
Total time to complete a story card
New lines of code/ development effort
LOC/person hour
LOC/development effort
User stories/ person hour
Effort/Time
Mean-time per assertion
Mean time
Total/overall time
Avg. hours worked
Time taken in person hours
Time taken on testing as a percentage of total time
Time spent on coding as a percentage of total time
Total person hours
Size
LOC per module
Sum of active and passive time
New LOC written during production
New LOC written during testing
Number of tests written
External Quality
Defect count
Defect density
Mean time assertion per method
Total assertions (passed/failed)
Number of test cases passed
Internal Code Quality
Method/statement/ condition coverage
Branch coverage
Nested block depth
Cyclomatic complexity
Number of parameters
Coupling between objects
Information flow
Weighted class per method
Lack of cohesion metrics
Mutation score indicator
Developer Opinion
Likert scale opinion
Exam scores
Questionnaires (on defects, correctness, simplicity, best choice)
Robustness
Web service deployment
Conformance
8 criteria selected from approved teaching staff
TDD changes and refactorings/all changes

Fig. 3. Themes, pattern and metrics used to measure TDD impact.

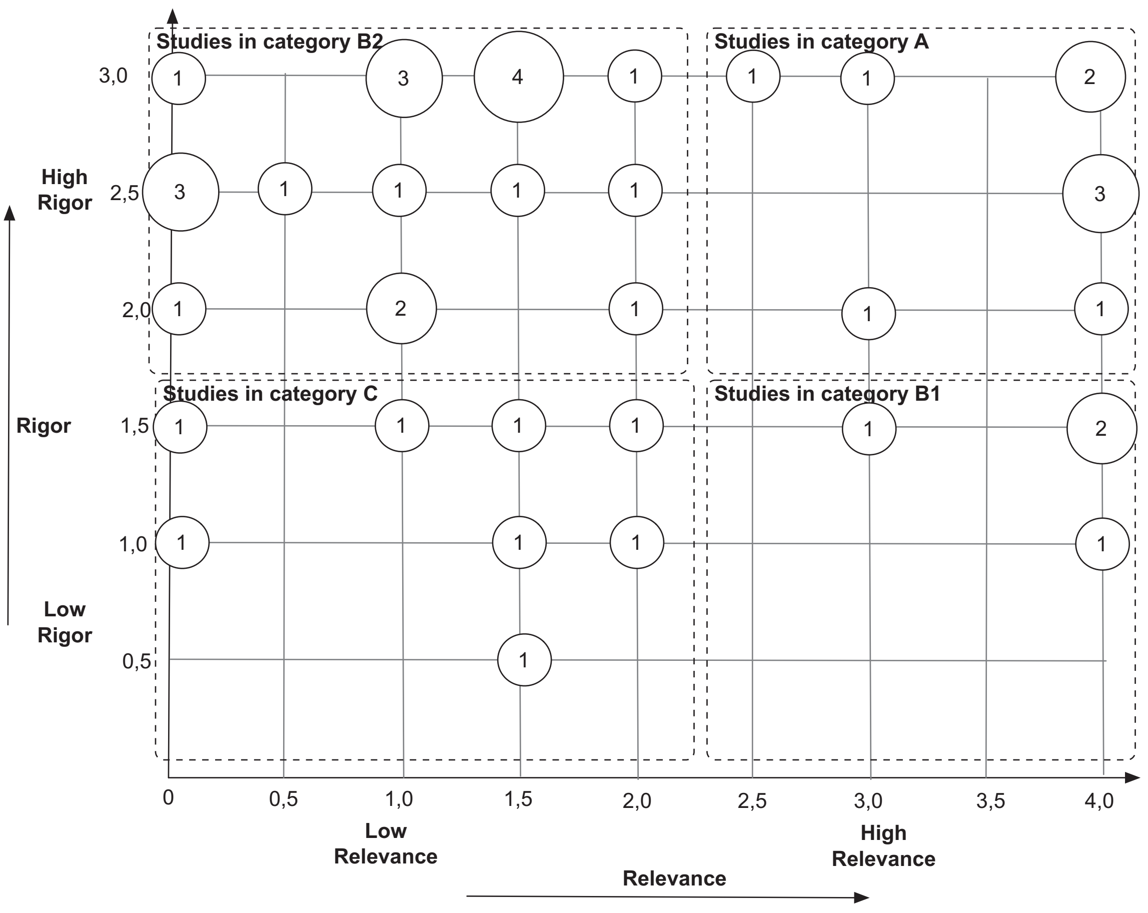

Fig. 4. Rigor and relevance grid.

Fig. 4 provides an overview of the individual studies and their scores for rigor and relevance.

To systematize the analysis, we defined categories for rigor and relevance.

- Low rigor is defined for a score of 0–1.5, while high rigor is defined for a score of 2.0 and above.
- Low relevance is defined for a score of 0–2.0 and as high for a score of 2.5 and above.

This provides a grid of combinations, A-studies being of high rigor and relevance, B-studies having either high relevance and low rigor (B1) or low relevance and high rigor (B2), and C-studies having low rigor and low relevance.

In category A we have 9 studies, which represents less than 20% of the total set of included primary studies. This indicates that overall the quality and reporting of the studies on TDD is only fair. In category B1 we have 4 studies, and B2 20 studies. Only 8 studies are found in category C.

4.2.1. A-studies: High rigor and high relevance studies

In category A we identified 7 case studies, and 2 surveys. Table 5 shows the results of the vote counting for studies in category A with respect to the outcome variables considered by studies in this category. We classify the outcome for the variables into: in favor of TDD, no difference, and negative for TDD.

With respect to productivity there is either no change in productivity, or a decline is observed. Overall, only few studies in category A are concerned with productivity.

Table 5
Outcome of studies (category A) with respect to variables.

Variable	In favor of TDD	No diff.	Neg. for TDD
Productivity		[64]	[59,60]
External quality	[74,60,59,63,62,71,64]		
Complexity	[61,71]		
Developer opinion	[72,74]		
Total	11	1	2

With respect to external quality we can see that studies are only positive for TDD, with 7 studies supporting this finding.

Two studies indicate that code complexity is reduced using TDD.

With respect to developer opinion the developers indicated positive results for TDD. In particular, in [72,74] the developers perceived an improvement in quality, further supporting the measured quality improvement documented for studies in category A. Furthermore, according to the developers no productivity was gained or lost. The details of the studies in category A can be found in Appendix B.

Conclusion: Based on the results above we draw the following conclusion for A-studies: Practitioners wanting to adopt TDD will improve their code quality as well as reduce complexity, and at the same time maintaining or loosing their productivity levels.

4.2.2. B1-studies: Low rigor and high relevance studies

In category B1 we identified 4 case studies. Table 6 the results of the vote counting are shown.

Table 6
Outcome of studies (category B1) with respect to variables.

Variable	In favor of TDD	No diff.	Neg. for TDD
External quality	[66–69]		
Time/effort			[66,68]
Performance/productivity			[67]
No. of studies	4	0	3

Studies in category B1 indicate positive results for external quality (4 studies are in favor, none against or indifferent). Time/effort is increased, as well as productivity reduced (only negative results reported for TDD in category B1). The details of the studies in category B1 can be found in Appendix C.

Conclusion: The vote counting would suggest that there is a potential to increase external quality, but at the expense of development time and productivity. Overall, only a low number of studies is found in this category.

4.2.3. B2-studies: High rigor and low relevance studies

In category B2 we identified 19 experiments and 1 case study. The results of the vote counting are illustrated in Table 7. With regard to productivity only one study shows results in favor of productivity, while all others show no difference. Related to that, effort/time shows conflicting results, with 4 studies reporting on time savings, while 2 studies indicate no difference.

Results for external quality are conflicting, with 3 studies supporting an improvement in external quality, 6 studies showing no difference, and 1 study indicating negative results. Internal code quality only has 1 study with positive results, 10 studies indicate no difference.

The vote counting shows that the number of tests is not different between TDD and TLD (4 studies), only 1 study indicates that fewer tests are written, and another that more tests are created.

Moreover, 3 studies were found addressing the impact TDD on the code size. Among them one studies resulted in no difference and the other study ended up with negative results when TDD is adopted, suggesting that TDD potentially increases the amount of code.

With regard to developer opinion, 4 studies show indifference to using TDD (i.e. the developers believe that using TDD instead of TLD does not bring significant benefits), and 2 studies indicate favorable results.

One study investigated process conformance and found positive results for TDD.

The details of the studies in category B2 can be found in Appendix D.

Conclusion: The majority of studies stipulated that there is no difference when TDD is compared with TLD for the variables mentioned in Table 7, while a small number of studies are in conflict (e.g. for external quality, or number of tests, as these have positive as well as negative results for the mentioned variables).

Table 7
Outcome of studies (category B2) with respect to variables.

Variable	In favor of TDD	No diff.	Neg. for TDD
Productivity	[46]	[49,5,6,41,42,45,51,54,53,48]	
External quality	[42,46,48]	[39,45,50,40,5,41]	[38]
Internal code quality	[46]	[49,39,56,50,54,6,41,43,45,48]	
No. of tests	[49]	[51–53,56]	[6]
Code size		[54,45]	[53]
Effort/time	[38,6,39,56]	[42,43]	
Developer opinion	[65,45]	[51,54,47,56]	
Conformance	[39]		
Total	13	38	3

Table 8
Outcome of studies (category C) with respect to variables.

Variable	In favor of TDD	No diff.	Neg. for TDD
Productivity			[73]
External quality	[73]	[3]	
Internal code quality		[44,3,52]	
Effort/time	[70,52]		
No. of tests		[44,52]	
Robustness		[57]	
Developer opinion	[77,13]		
Code size	[44]		
Total	6	7	1

4.2.4. C-studies: Low rigor and low relevance studies

Category C comprised of 6 experiments, 1 survey, and 1 case study. Results of the vote counting are shown in Table 8.

Only one individual study in category C investigated productivity with negative results. On the other hand, effort/time savings have been reported by 2 other studies.

External quality shows a potential improvement with 1 study in favor, and one showing no difference. For internal code quality, no difference could be demonstrated.

Two studies investigated the number of test cases with the result that there is no difference between TDD and TLD. The code size has been reduced, showing positive results for TDD.

Developer opinion was positive for the use of TDD, supported by two studies.

Only one study was found that investigated robustness and stated that TDD does not introduce any difference in the robustness compared to TLD. Moreover, it is to be noticed that one study [58] was found that investigated productivity and code quality but did not present results of the study. The details of the studies in category C can be found in Appendix E.

Conclusion: Studies disagree on whether there is a positive or no effect when using TDD, only one study shows negative results.

5. Discussion

5.1. Variables

Based on the studies found in this systematic literature review it is of prime importance to identify the right set of variables in order to discover the positive or negative impacts of TDD compared to TLD. Various research methodologies have been used to investigate the effect of TDD over TLD such as case studies, experiments, surveys and systematic literature reviews. In case of experiments, there are a number of themes or patterns that have been identified as can be seen in Fig. 3. However, the majority of the studies lacked consistency in terms of their variables that have been used to examine the positive or negative impacts of TDD. Moreover, if the variables are found consistent in some studies then their units of measures and measurement criteria were not consistent enough to aggregate those studies. It is also noticed that experimental studies have not reported their statistical values, which makes generalization of result extremely difficult through statistical meta analysis techniques. Similar findings were obtained by [14] who conducted a meta-analysis, stating that: *“First, since the metrics used to operationalize the outcome constructs differ from study to study, (. . .), 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."* Madeyski [78] provided a set of experiments and conducted a meta-analysis of them. The reporting of these experiments could be used as a guideline to improve the reliability of meta-studies. Furthermore, Madeyski [78] also reviewed outcome variables.

Therefore, it is important for researchers to identify already investigated variables in TDD studies and agree on a set of variables and how to measure them. That way they are consistent across future studies and can be aggregated without additional validity threats. The right set of variables could be external code quality, internal code quality and productivity, since the majority of the studies have investigated these variables to identify the positive or negative impacts of TDD compared to TLD. Furthermore, it is also of importance to report all statistical values (p-values, effect size, mean, median, standard deviation and confidence interval) when testing hypotheses. This will enable researchers later, particularly through the use of meta analysis, to conclude whether or not TDD delivers better performance compared to TLD. In future meta-analysis on TDD we suggest to contact authors to obtain the complete statistical information This might also help identifying studies that were conducted, but not published due to negative results.

An important and challenging to measure variable is conformance to the process. Conformance is checked in a semi-automatic way using plug-ins of the integrated development environment (cf. [50,53]). It would be desirable to use source triangulation (e.g. observations, questionnaires) to complement, and hence increase the reliability of the findings, given that it might be a large confounding factor in outcomes. Overall, only few studies investigated conformance. Lack of conformance has been highlighted as an important confounding factor by [14].

5.2. Rigor and relevance outcomes

We categorized studies in A (high rigor and high relevance), B1 (low rigor and high relevance), B2 (high rigor and low relevance), and C (low rigor and low relevance). For each category the conclusions are different when looking at each set individually. We shortly repeat the main conclusions for each set:

- A: Practitioners wanting to adopt TDD will improve their code quality as well as complexity and at the same time maintain or lose their productivity levels.
- B1: There is a potential to increase external quality, but at the expense of development time and productivity.
- B2: The majority of studies stipulated that there is no difference when TDD is compared with TLD for the variables identified.
- C: Studies disagree on whether there is a positive or no effect when using TDD, only one study shows negative results.

The conclusions in categories A and B1 are relatively similar. Both studies find that external quality is improved, and no other study in these categories contradicts that finding. Both categories also agree on productivity, in particular that there is a risk of loosing productivity. However, only very few results are available for productivity in these categories. Also noteworthy is that the variable set in categories A and B1 is relatively small, as it primarily focuses on external quality and productivity, while not focusing on, for example, internal code quality.

Studies in category B2 are different from those in categories A and B1. B2 studies are of high rigor and received low relevance scores. A higher number of variables has been assessed compared to A and B1, in particular many studies investigated internal code quality. Overall, the majority of studies in B2 would indicate that there is no difference when using TDD. For example, for external quality the majority of studies found no difference, while there were also studies showing positive as well as negative effects. That is, would one combine studies in categories A and B1 with studies in category B2 the result would become inconclusive with many results being in favor, showing no difference, and being negative for TDD.

Studies in category C show a balanced result of either achieving an improvement, or no difference when using TDD. Studies in this category would support conclusions from studies in category B2 and bias the result in favor of TDD when combining the whole set.

Given that the conclusions are different between the categories, and when combining studies the conclusions become inconclusive, we found that using a more fine-grained scale for both, rigor and relevance, is important when analyzing empirical research results. This emphasizes the relevance of the instrument proposed by Ivarsson and Gorschek [12].

5.3. Comparing with previous literature reviews

Tables 9–11 provide an overview of the distribution of results for TDD in the different literature reviews. Literature reviews that are comparable, due to the use of vote counting, are Shull et al. [9] and Kollanus [11]. Shull et al. distinguished sets of studies based on quality, hence the distribution is shown for the sets of all studies and high quality studies.

In Table 9 (external quality) we can see that there is a difference between all literature reviews and studies classified in category A and B1. Using the assessment instrument by Ivarsson and Gorschek [12] the result of high rigor and high relevance studies is clearly positive for external code quality for studies in categories A and B1. One potential reason can be the extension of definition of the relevance dimension and the consideration of rigor, both on a more fine-grained scale. The distribution of studies in categories B2 and C is closer the distributions found in the previous reviews. Looking at the totals of the studies we can see that the distributions between the studies for the overall number of studies are very similar indicating high reliability of the results and also supporting generalizability.

Table 10 provides an overview of internal code quality, where the studies are not as evenly distributed between the outcomes as is the case for the literature reviews. For A all studies are in favor (only code complexity has been considered in category A). For the totals our study is well aligned with the distribution given by Kollanus. For the variation a potential source of difference could be the definition of variables (e.g. we included code coverage as an internal quality attribute) and the inclusion of studies published after Kollanus and Shull et al. finalized their reviews. According to previous literature reviews the results seem to be inconclusive, while in our review too few studies investigated internal code quality in categories A and B1 to draw conclusions.

Table 11 shows the results of for productivity. Overall, for A-studies it is found that no positive results have been obtained.

Table 9
Comparison with related work (external quality), in %.

Study	In favor of TDD	No diff.	Neg. for TDD
Shull all	62	29	9
Shull quality	39	46	15
Kollanus	72	23	5
A	100	0	0
B1	100	0	0
B2	30	60	10
C	50	50	0
Total	65	30	5

Table 10
Comparison with related work (internal code quality), in %.

Study	In favor of TDD	No diff.	Neg. for TDD
Shull all	46	29	25
Shull quality	28	44	28
Kollanus	38	56	6
A	100	0	0
B1	0	0	0
B2	7	86	7
C	25	75	0
Total	20	75	5

Table 11
Comparison with related work (productivity, in %).

Study	In favor of TDD	No diff.	Neg. for TDD
Shull all	40	24	36
Shull quality	38	38	25
Kollanus	22	30	48
A	0	33	67
B1	0	0	100
B2	29	71	0
C	67	0	33
Total	27	50	22

No study in category B1 focused on productivity. All other literature reviews indicate inconclusive results with studies in all categories. On B2 the results indicate that there is no difference. In total the reviews show balanced results when not splitting them into categories, while the emphasis is slightly different. Shull finds more studies in favor, Kollanus more studies that are negative, and our review identifies the majority of studies in the category with no difference. This indicates that, in some cases, productivity might have been interpreted in different ways. For instance, in this aggregated view we also incorporated time/effort in the vote counting when comparing with the other studies.

Sfetsos and Stamelos [10] looked at internal and external quality in TDD. Their evaluation is in-line with Kollanus, also indicating varying results for internal as well as external quality (see above discussion for Kollanus).

Overall, by distinguishing studies based on rigor and relevance, it became visible that results depend on rigor and relevance, and hence show different results to previous literature reviews. Therefore, considering rigor and relevance in TDD evaluation is an important complement to previous literature reviews.

5.4. Effect of studies identified post-completion

As discussed in Section 3.5 with the completion of this review the final set of papers has been compared with other relevant reviews for our work. Ten additional studies have been identified [79–88].

Table 12 provides an overview of the additional articles. Six of them are in category C, and four in category B2.

The results show that for studies in category C (see Table 8) the result for external quality would be positively influenced by adding the studies from Table 12, and internal quality would receive a more balanced result. The reason for the additional studies being in category C is that they have been shorter papers (2–6 pages), hence being restricted in reporting everything related to the research design. Given the positive results for external and internal code quality of studies in B2, the result for that variable would become more balanced.

5.5. Validity threats

Gencel and Petersen [89] define five general classes of validity, independently of study type, namely descriptive validity, theoretical validity, generalizability, interpretive validity, and repeatability.

5.5.1. Theoretical validity

Theoretical validity is concerned with the potential presence of confounding factors and whether we are capable of capturing what we intend to capture.

One potential threat is publication bias affecting our results without our knowledge, i.e. primarily positive results have been published. Given that we are using vote counting, publication bias would have an effect on the results. However, given that we found a high number of studies showing no difference, or negative results, publication bias does not appear to be a threat in the context of TDD studies.

The scoring system by Gorschek and Ivarsson does not favor experiments, given that they tend to receive lower scores on the relevance dimension. Experiments have strengths in theoretical validity by controlling variables, but have limitations in external validity due to the laboratory setting and limited time frames. Therefore, it is a trade-off between theoretical validity (favoring experiments) and external validity (favoring industry studies). The intention of the scoring system proposed by Ivarsson and Gorschek is to rate the degree of realism, while there are limitations in considerations related to theoretical validity.

Table 12
Articles identified post-completion.

Ref.	Cat.	Rigor	Relevance	Pos. for TDD	No diff.	Neg. for TDD
[79]	C	0.5	0	Productivity; Int. code quality Ext. quality	–	–
[80]	C	0	0	Ext. quality; Dev. opinion	–	–
[81]	C	0.5	0	Ext. quality, Int. code quality	–	–
[82]	B2	2	1	Ext. code quality, Int. code quality, Dev. opinion	–	Time/effort
[83]	B2	2	0	External quality, development time, internal code quality, programmer opinion, size	–	–
[84]	B2	2.5	1	Programmer opinion (positive for external quality, internal quality, development speed)	Programmer opinion (complicated to follow)	–
[85]	B2	2	2		Developer opinion (external quality)	Developer opinion (usability, applicability for the specific domain, mobile)
[86]	C	0.5	0	External quality	Productivity (positive as well as negative results obtained)	Size, time
[87]	C	1	0	External quality, productivity		
[88]	C	1.5	0	Productivity, Internal code quality, programmer opinion	Complexity	

One potential threat to theoretical validity is a small sample size in experiments. The meta-analysis by [14] included 26 studies, where the average number of subjects was 46, the standard deviation was 49, the minimum number of subjects was 8 and the maximum number 166. Only 4 studies were having less than 10 subjects. The study also found small effect sizes, which could potentially be attributed to sample sizes, and also affects the results of this study given that there is a large overlap in publications. However, no a priori statistical power analysis has been conducted in the included experiments.

5.5.2. Generalizability

Generalizability is the ability to generalize the results to different settings, groups, and situations.

In terms of experiments the overall number of publications is high. Consequently, TDD has been studied in many different experimental contexts with different students, and at different institutions. Hence, one can make the proposition that for student experiments the conclusions may possibly not change when adding further experiments. Hence, we believe that the threat of generalizability for experiments (primarily being in category B2) is low.

However, the overall number of case studies in categories A and B is relatively low, which indicates that further studies are needed to substantiate the findings of this review.

5.5.3. Objectivity

Objectivity is the ability to describe what has been observed truthfully. In order to reduce this threat multiple actions have been taken.

First, there is the risk that the researchers miss studies in the selection process. This threat has been reduced by not relying on a single researcher in the inclusion/exclusion of studies. Instead, the researchers conducted the selection independently and disagreements were discussed and resolved. Furthermore, inclusion and exclusion criteria have been piloted and discussed.

Second, in the data extraction activity there is always a risk to miss or misinterpret information from the text. In order to reduce this threat, a data extraction form was designed and discussed among the authors to develop a shared understanding. The extraction was done by the first and second author, who reviewed each others extractions. Thereafter, the third author conducted a review of the extraction and interpretation of the results.

Third, the scoring of rigor and relevance, which is central in this article, is dependent on interpretation as well. As mentioned earlier, the scoring system was designed in the form of a rubric known in education. Evidence shows that rubrics make evaluation criteria transparent and bring consistency in the grading process [35–37], which increases confidence in the results. The scoring was done by the first and second author individually, and has been reviewed by the third author.

Fourth, the definitions of the variables (due to heterogeneity of variable definition in general) might not be completely aligned between studies on individual variables. The general categorization shows that authors of the different studies have a similar understanding, but potential different understandings of the definitions remains a threat.

Overall, actions have been taken to reduce threats to objectivity, however, we acknowledge that it is not possible to mitigate them as they rely on interpretation.

5.5.4. Interpretive validity

Interpretive validity is concerned with drawing the right conclusions from the given data in an objective manner.

In this study we followed the vote counting approach, and the rigor and relevance criteria [12]. In particular, details provided in [12] highlight and clarify how to interpret and judge rigor and relevance criteria, which aids in drawing the conclusions.

5.5.5. Repeatability

Repeatability is concerned with a clear definition of instruments, hence aiding in the repeatability. That is, another researcher following the protocol should come to similar conclusions.

The repeatability is positively affected by following systematic review guidelines [21,22] and guidelines for quality assessment [12]. We developed a review protocol, and provided details of the study in Section 3 to enable other researchers to extend the review in the future.

6. Conclusion

We conducted a systematic literature review on TDD in comparison to TLD. The review takes rigor and relevance into account. Rigor documents the extent to which study context, design, and validity are described. Relevance determines to what degree the results have impact on industry practice in terms of realistic subjects, scale, research context, and research method focusing on realistic environments.

We investigated which outcome variables and units of measure were used to capture TDD performance, extracted variables and actual outcomes in studies relating them to rigor and relevance evaluation, and contrasted the findings with previous systematic reviews. Our findings are:

1. We identified 8 variables that have been considered when evaluating TDD. However, each variable has been quantified in different ways, using different units of measures. This makes aggregation of studies (e.g. through meta-analysis) cumbersome and yields a validity threat in aggregation.
2. After classifying studies based on the rigor and relevance levels in four categories based on high/low rigor and high/low relevance, we looked at each category in isolation and drew conclusions. Only 9 studies are in the high rigor and high relevant category, which indicates that the reporting completeness of studies on TDD is only fair. We found that in each category different conclusions were obtained, hence the results would be skewed when looking at the studies as a complete set. Studies in the high rigor and high relevance category (in total 9 studies) show positive results for external quality, and indicate that no difference or negative results are obtained for productivity. A similar observation was made by studies with high relevance and low rigor. Studies with low relevance and high rigor mostly show no difference with respect to a larger set of variables, while categories that received low relevance and rigor scores show a balanced picture for positive results and no difference in results. Looking at these studies as one set, the results would be inconclusive. Interestingly, as is the case for studies in category A, the positive results for quality are not reflected in time savings due to better quality when looking at the outcomes. A potential reason could be that long-term observations are required across multiple releases to see the benefits, further underlining the need for longitudinal studies.
3. When comparing this result with existing literature reviews on TDD, different conclusions have been reached. In particular, study results do not vary as much compared to previous reviews for studies with high relevance ratings.

Based on these conclusions we provide the following recommendations for future work:

- Studies are judged based on what is reported, which is what the rigor and relevance framework by Ivarsson and Gorschek emphasizes. Therefore, we highlight the need to arrive at a more complete reporting of context and validity threats.
- Based on high relevance and high rigor studies we can conclude that TDD has positive effects on external quality, however, these results have to be further substantiated with a larger number of future studies in that category. In future work there is a need for further empirical studies on TDD, preferably industrial case studies taking a long-term perspective, and industrial experiments with real-world systems.
- Given that high quality and relevance studies focused primarily on external quality, the other variables (e.g. internal quality, productivity, number of tests, or process conformance) need further evaluation.
- Experiments only allow to capture a snapshot, and short period of development and execution of test cases. In order to better understand the effect of TDD, more long-term observations are needed.

Future work should focus on conducting comparisons with the aim of designing and reporting studies so that they would be located in category A, as the overall number of studies in that category was limited.

Acknowledgments

The work has been funded by ELLIIT, the Strategic Area for ICT research, funded by the Swedish Government. We would also like to thank Jürgen Börstler for valuable feedback on the study.

Appendix A. Rigor and relevance of primary studies

Rigor defines the degree to which context (C), study design (D), and validity threats (V) have been described [12]. The values for individual studies are provided in Table A.13.

Relevance describes the degree to which the studied setting is realistic from an industry practice perspective [12]. Studies were rated for users (U), scale (S), research methodology (RM), and context (C) (see Table A.14).

Appendix B. Summary of studies in category A

Table B.15.

Table A.13
Studies scored based on rigor.

ID	Reference	C	D	V	C+D+V
1	[72]	1	0.5	0.5	2
2	[60]	1	1	1	3
3	[64]	1	1	0.5	2.5
4	[65]	1	1	1	3
5	[43]	1	1	1	3
6	[73]	0	0.5	0	0.5
7	[48]	1	0.5	0.5	2
8	[59]	1	1	1	3
9	[62]	1	1	1	3
10	[77]	1	0.5	0	1.5
11	[6]	1	1	1	3
12	[70]	0.5	0.5	0.5	1.5
13	[52]	1	0.5	0	1.5
14	[56]	0.5	1	1	2.5
15	[49]	1	0.5	0.5	2
16	[46]	1	1	0.5	2.5
17	[42]	1	1	1	3
18	[58]	1	0.5	0.5	2
19	[5]	1	1	1	3
20	[47]	1	1	0.5	2.5
21	[44]	0.5	0.5	0.5	1.5
22	[54]	1	1	0.5	2.5
23	[51]	1	1	0.5	2.5
24	[68]	0.5	0.5	0.5	1.5
25	[13]	0.5	0.5	0	1
26	[57]	0.5	0.5	0	1
27	[69]	1	0.5	0	1.5
28	[40]	1	1	1	3
29	[39]	1	1	0.5	2.5
30	[53]	1	0.5	0.5	2
31	[50]	1	1	1	3
32	[66]	0.5	0.5	0.5	1.5
33	[41]	1	1	1	3
34	[3]	0.5	0.5	0	1
35	[61]	1	1	0.5	2.5
36	[71]	1	0.5	0.5	2
37	[74]	1	1	0.5	2.5
38	[63]	1	1	1	3
39	[45]	1	1	0.5	2.5
40	[38]	1	1	1	3
41	[67]	0.5	0.5	0	1

Table A.14
Studies scored based on relevance.

ID	Reference	U	C	RM	S	Sum
1	[72]	1	1	1	0	3
2	[60]	0.5	1	1	0	2.5
3	[64]	1	1	1	1	4
4	[65]	0.5	0	1	0	1.5
5	[43]	0	0	1	0	1
6	[73]	0.5	0	1	0	1.5
7	[48]	0	0	1	0	1
8	[59]	1	1	1	1	4
9	[62]	1	1	1	1	4
10	[77]	0	0	1	0	1
11	[6]	0	0	1	0	1
12	[70]	0	0	0	0	0
13	[52]	0.5	0	1	0	1.5
14	[56]	0.5	0	0	0	0.5
15	[49]	1	0	1	0	2
16	[46]	1	0	1	0	2
17	[42]	0.5	0	0	1	1.5
18	[58]	0	0	0	0	0
19	[5]	0	1	1	0	2
20	[47]	0	0	0	0	0
21	[44]	1	0	1	0	2
22	[54]	0	0	0	0	0
23	[51]	0	0	0	0	0
24	[68]	1	1	1	0	3
25	[13]	0.5	0	1	0	1.5
26	[57]	1	0	1	0	2
27	[69]	1	1	1	1	4
28	[40]	0.5	0	1	0	1.5
29	[39]	0.5	0	1	0	1.5
30	[53]	1	0	0	0	1
31	[50]	0.5	0	1	0	1.5
32	[66]	1	1	1	1	4
33	[41]	0	0	0	0	0
34	[3]	0	0	0	0	0
35	[61]	1	1	1	1	4
36	[71]	1	1	1	1	4
37	[74]	1	1	1	1	4
38	[63]	1	1	1	0	3
39	[45]	0	0	1	0	1
40	[38]	1	0	0	0	1
41	[67]	1	1	1	1	4

Table B.15
Summary of studies in category A.

Study	Method	Subject/context	Variables	Outcome
[59]	Case study	Three commercial projects	Productivity, external code quality	TDD developers produced higher quality code, but reduction in productivity
[60]	Case study	Three software development projects	External code quality, productivity	TDD increased 2.64.2 times compared to non-TDD developed code. Significant defect rate reduction in the new project using TDD; the defect density was reduced in all the studies conducted in industry. However, the productivity reduced as it took 16% more time
[61]	Case study	Five small scale software development projects. Two of the projects used TLD development and three utilized TDD	Complexity	Statistically significant differences in cyclomatic complexity. TDD helps produce less complex code
[63]	Case study	A framework of reusable components in place in the IT-department of a large Norwegian Oil and Gas company. Microsoft excel was used as a tool to analyze changes for the five releases of the reusable JEF framework	External code quality	TDD yields fewer defects overall
[72]	Survey	218 volunteer programmers	Programmer opinion	A 40–50% reduction in defect density, passed in up to 50% more external tests than code produced by control groups not using TDD. Minimal impact to productivity when programmers in industry
[62]	Case study	100 professionals	External code quality	The researchers observed that TDD reduced fault slip-through by 530%, and avoidable fault costs by 55%
[71]	Case study	Students and professionals, 5 long-term open source projects	Complexity	The first 2 projects used iterative test last and remaining 3 used TDD. Results revealed that the 3 projects that used TDD approach wrote significantly less complex code
[74]	Survey	The project is the development of IBMs JavaPOS 3 -compliant device drivers. The project consists of the creation of middleware for devices in the POS domain. The development team initially consisted of nine full-time engineers, five in Raleigh, NC, USA and four in Guadalajara, Mexico	Programmer opinion	Both the external and the internal defect density across all releases are significantly lower than the industry averages. developers indicated that the use of TDD got easier over time: The developer avoided running the manual tests and spending the time to review the output of the tests
[64]	Case study	Study of two units of analysis (divisions of microsoft)	External code quality, productivity	The productivity of the team was not impacted by the additional focus on producing automated test cases. 40% fewer defects than a more traditional fashion

Appendix C. Summary of studies in category B1

Study	Method	Subject/context	Variables	Outcome
[66]	Case Study	3 development teams at Microsoft and IBM. The device drivers on a new platform, Windows, MSN, Developer Division	External code quality, time/effort	The pre-release defect density of the four products decreased between 40%, the teams experienced a 1535% increase in initial development time after adopting TDD. The increase in time to develop the features attributed to the usage of the TDD practice
[67]	Case Study	IBM; All participating IBM software engineers on both projects had a minimum of a bachelor's degree in computer science, Two had master's degrees Developing device drivers for over a decade	External code quality, Productivity	Defect rate significantly better for the new system compared with the legacy system. Slight decrease in developer productivity when employing the TDD practice
[68]	Case Study	A cost accountant with in excess of 10 years experience in the area, an experienced software developer with an expert knowledge of TDD, a college Mathematics lecturer with an intermediate knowledge of spreadsheet development but no experience of TDD	External code quality	Test-Driven Developers produced a higher quality of code passed 18% more functional black box test cases. 80% of the developers believed TDD was more effective. Increased development time as he had to create a group of tests rather than one single test without multiple assertions
[69]	Case study	IBM RSS group, JavaPOS project	External code quality	50% reduction in defect density

Appendix D. Summary of studies in category B2

Study	Method	Subject/context	Variables	Outcome
[50]	Experiment	Master students	Internal code quality	No statistical significant difference was found between TDD and TLD team regarding branch coverage (BC), Mutation score indicator (MSI)

Appendix D. (*continued*)

Study	Method	Subject/context	Variables	Outcome
[65]	Case study	Eight programmers who had just adopted extreme programming	Programmer opinion	Report on positive experience with respect to external quality and resulting customer satisfaction
[54]	Experiment	Undergraduate students	Productivity, code size, internal code quality, developer opinion	Little or no difference in terms of productivity, code size, internal code quality and programmers opinion
[53]	Experiment	Professional	Productivity, number of tests	Programmer productivity was higher in TDD (LOC, user stories) but cannot be considered as significant. The same applies to the number of tests written
[38]	Experiment	Computer science students (SP)	Automated acceptance testing, total man hours	TDD is less effective at reducing bugs, but more time efficient (compared with manual static verification through inspection)
[40]	Experiment	Master students (SP,PP)	External code quality	Little or no significant difference in terms of acceptance test cases passed
[5]	Experiment	Undergraduate students (SP)	Productivity, external quality	Productivity in TDD team was 70% higher, but not statistically significant. TDD team spent significantly more time on testing. Little or no difference in external quality
[6]	Experiment	Undergraduate students (SP)	Number of tests, internal code quality, productivity	TD programmers wrote significant more tests. TD programmers produced have little or no significant difference in productivity and internal code quality (test quality). Higher number of tests imply higher productivity was also rejected
[41]	Experiment	Undergraduate students (SP, PP)	Productivity, external quality, internal code quality	No statistical difference regarding productivity, external quality, and internal code quality (acceptance test and mutation score indicator)
[42]	Experiment	Students (SP)	external code quality, development effort, developer productivity	Significant in external code quality. However, little or no difference in development effort and productivity
[43]	Experiment	Professionals (SP)	Effort/time, internal code quality	No evidence that TDD requires more time than TLD. Internal code quality with respect to written tests shows no evidence that TDD enables more accurate and precise test cases
[45]	Experiment	Undergraduate students (SP)	Productivity, code size, internal code quality, external code quality, developer opinion	Little or no difference between found in terms of productivity, LOC, internal code quality and external code quality. 10 out of 14 subjects preferred TDD over TLD
[51]	Experiment	Students (SP)	Number of tests, productivity, developer opinion	Little or no significant difference were found in case of number of tests, productivity, and developer opinion
[39]	Experiment	Computer science students (SP)	Conformance, speed/effort, external code quality, internal code quality size	Experts achieved significant conformance and shorter test cycles. A novice who is able to follow the TDD process changes the test code significantly more than a novice who is not able to apply TDD. Experts have high editing speed. The experts reach to first acceptance test earlier than novices. Little or no differences in terms of external quality (failed tests) and internal code quality (test coverage)
[46]	Experiment	Professionals (PP)	External code quality, productivity, internal code quality	TDD showed significant external quality, productivity and internal code quality (branch coverage)
[47]	Experiment	Undergraduate students (SP)	Developer opinion	Result was not significant enough to conclude that test subjects prefer-test first approach
[49]	Experiment	Professionals (SP)	Productivity, number of test cases, internal code quality	Little or no difference in terms of productivity and internal code quality (branch coverage). The number of test cases was higher in TDD

(*continued on next page*)

Appendix D. (*continued*)

Study	Method	Subject/context	Variables	Outcome
[48]	Experiment	Undergraduate and graduate students (SP)	Productivity, external quality, internal code quality	TDD group passed more unit tests than TLD group. However, no significant difference in productivity and internal code quality
[56]	Experiment	Graduate students	Effort/time, number of test cases, internal code quality (test coverage), developer opinion	Development speed in TDD was faster in developing story cards, there was no difference in number of test cases, over time no significant differences between TDD and TLD in coverage, no preference for TDD or TLD given by the subjects
[58]	Experiment	Undergraduate students (SP)	Productivity, code quality	Only the experimental set-up is described thoroughly, not the results of the investigation

Appendix E. Summary of studies in category C

Study	Method	Subject/context	Variables	Outcome
[70]	Case study	An open-source Java library (JFreeChart (version1.0.9)) to develop professional quality charts	Effort/time	Time savings in debugging have been reported
[57]	Experiment	Professionals (SP)	Robustness	TDD is practical way of preventing the robustness problems in the deployment of web services
[73]	Survey	The most widely used language for TDD is Java, along with Junit	External quality, productivity	35% and 45% in some areas reduction in defects. Others found less hopeful results of a 510% decrease in productivity
[44]	Experiment	Professionals (SP)	Code size, Internal code quality	TDD programmers write significantly smaller methods on average (code size). However, little or no difference found in terms of internal code quality (improved cohesion)
[77]	Experiment	Students	Developer opinion	Higher satisfaction achieved through pair programming with the TDD approach
[3]	Experiment	Undergraduate students	External quality, Internal code quality	The average mean of code coverage is 92.6% for TDD and 95.1% for TLD. TDD team passed fewer test cases than TLD. However, differences are small and not significant
[13]	Experiment	Master students	Developer opinion	Students with no prior TFD experience had more difficulties with TDD. 20 pairs out of 27 were willing to use TDD in future
[52]	Experiment	Graduate students	Internal code quality, number of tests, effort/speed	Little or no significant difference in terms of development speed, number of test cases, and internal code quality (code coverage)

References

[1] K. Beck, Extreme Programming Explained: Embrace Change, second ed., 2004

[2] M. Fowler, Refactoring: Improving the Design of Existing Code, Addison-Wesley Professional, 1999.

[3] M. Pancur, M. Ciglaric, M. Trampus, T. Vidmar, Towards empirical evaluation of test-driven development in a university environment, Proceedings of IEEE Region 8 EUROCON 2003, Computer as a Tool, 22–24 September 2003, vol. 2, IEEE, Piscataway, NJ, USA, 2003, pp. 83–86.

[4] H. Cibulski, A. Yehudai, Regression test selection techniques for test-driven development, in: 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation Workshops (ICSTW), IEEE, 2011, pp. 115–124.

[5] L. Huang, M. Holcombe, Empirical investigation towards the effectiveness of test first programming, Inform. Softw. Technol. 51 (1) (2009) 182–194.

[6] H. Erdogmus, M. Morisio, M. Torchiano, On the effectiveness of the test-first approach to programming, IEEE Trans. Softw. Eng. 31 (3) (2005) 226–237.

[7] S. Kollanus, Test-driven development – still a promising approach?, in: 7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010, September 29, 2010–October 2, 2010, IEEE Computer Society, 2010, pp 403–408.

[8] A. Causevic, D. Sundmark, S. Punnekkat, Factors limiting industrial adoption of test driven development: a systematic review, in: Proceedings 2011 IEEE Fourth International Conference on Software Testing, Verification and Validation (ICST 2011), Los Alamitos, CA, USA, 2011, pp. 337–46.

[9] F. Shull, G. Melnik, B. Turhan, L. Layman, M. Diep, H. Erdogmus, What do we know about test-driven development?, IEEE Softw 27 (6) (2010) 16–19.

[10] P. Sfetsos, I. Stamelos, Empirical studies on quality in agile practices: a systematic literature review, in: 7th International Conference on the Quality of Information and Communications Technology, QUATIC 2010, September 29, 2010–October 2, 2010, IEEE Computer Society, Porto, Portugal, 2010, pp. 44–53.

[11] S. Kollanus, Critical issues on test-driven development, in: 12th International Conference on Product-Focused Software Process Improvement, PROFES 2011, June 20, 2011–June 22, 2011, LNCS, vol. 6759, Springer Verlag, Torre Canne, Italy, 2011, pp. 322–336.

[12] M. Ivarsson, T. Gorschek, A method for evaluating rigor and industrial relevance of technology evaluations, Empir. Softw. Eng. 16 (3) (2011) 365–395.

[13] S. Kollanus, V. Isomottonen, Test-driven development in education: experiences with critical viewpoints, in: Proceedings of the Conference on Integrating Technology into Computer Science Education, ITiCSE, June 30, 2008–July 2, 2008, Association for Computing Machinery, Madrid, Spain, 2008, pp. 124–127.

[14] Y. Rafique, V. Misic, The effects of test-driven development on external quality and productivity: a meta-analysis, IEEE Trans. Softw. Eng. 39 (6) (2013) 835–856.

[15] R. Jeffries, G. Melnik, Guest editors' introduction: TDD–the art of fearless programming, IEEE Softw. 24 (3) (2007) 24–30.
[16] K.S. Khan, G. Ter Riet, J. Glanville, A.J. Sowden, J. Kleijnen, et al., Undertaking Systematic Reviews of Research on Effectiveness: CRD's Guidance for Carrying Out or Commissioning Reviews, second ed., vol. 4, NHS Centre for Reviews and Dissemination, 2001.
[17] T. Greenhalgh, How to Read a Paper: The Basics of Evidence-based Medicine, third ed., Blackwell, Malden, Mass., 2006.
[18] B. Turhan, L. Layman, M. Diep, H. Erdogmus, F. Shull, How effective is test-driven development, Making Software: What Really Works, and Why We Believe It, 2010, pp. 207–217.
[19] T. Greenhalgh, R. Peacock, Effectiveness and efficiency of search methods in systematic reviews of complex evidence: audit of primary sources, BMJ 331 (7524) (2005) 1064–1065.
[20] S. Jalali, C. Wohlin, Systematic literature studies: database searches vs. backward snowballing, in: 2012 ACM-IEEE International Symposium on Empirical Software Engineering and Measurement (ESEM 2012), 2012, pp. 29–38.
[21] B. Kitchenham, O. Pearl Brereton, D. Budgen, M. Turner, J. Bailey, S. Linkman, Systematic literature reviews in software engineering – a systematic literature review, Inform. Softw. Technol. 51 (1) (2009) 7–15.
[22] K. Petersen, N.B. Ali, Identifying strategies for study selection in systematic reviews and maps, in: Proceedings of the 5th International Symposium on Empirical Software Engineering and Measurement, ESEM 2011, Banff, AB, Canada, September 22–23, 2011, pp. 351–354.
[23] B. Kitchenham, P. Brereton, D. Budgen, Mapping study completeness and reliability - a case study, in: Proceedings of the 16th International Conference on Evaluation & Assessment in Software Engineering (EASE 2012), 2012, pp. 126–135.
[24] C. Wohlin, R. Prikladnicki, Systematic literature reviews in software engineering, Inform. Softw. Technol. 55 (6) (2013) 919–920.
[25] J.E. Hannay, T. Dybå, E. Arisholm, D.I.K. Sjøberg, The effectiveness of pair programming: a meta-analysis, Inform. Softw. Technol. 51 (7) (2009) 1110–1122.
[26] T. Dybå, T. Dingsøyr, G.K. Hanssen, Applying systematic reviews to diverse study types: an experience report, in: Proceedings of the First International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, Madrid, Spain, September 20–21, 2007, pp. 225–234.
[27] H. Kundel, M. Polansky, Measurement of observer agreement, Radiology 228 (2) (2003) 303.
[28] C. Wohlin, P. Runeson, P.A. da Mota Silveira Neto, E. Engström, I. do Carmo Machado, E.S. de Almeida, On the reliability of mapping studies in software engineering, J. Syst. Softw. 86 (10) (2013) 2594–2610.
[29] C. Wohlin, P. Runeson, M. Höst, M.C. Ohlsson, B. Regnell, A. Wesslén, Experimentation in Software Engineering, Springer, Berlin, 2012.
[30] P. Runeson, M. Höst, Guidelines for conducting and reporting case study research in software engineering, Empir. Softw. Eng. 14 (2) (2009) 131–164.
[31] K. Petersen, C. Wohlin, Context in industrial software engineering research, in: Proceedings of the Third International Symposium on Empirical Software Engineering and Measurement, ESEM 2009, Lake Buena Vista, Florida, USA, October 15–16, 2009, pp. 401–404.
[32] D.I.K. Sjøberg, T. Dybå, M. Jørgensen, The future of empirical methods in software engineering research, in: Workshop on the Future of Software Engineering (FOSE 2007), 2007, pp. 358–378.
[33] M.V. Zelkowitz, D.R. Wallace, D. Binkley, Culture conflicts in software engineering technology transfer, in: NASA Goddard Software Engineering Workshop, 1998, p. 52.
[34] D.I.K. Sjøberg, J.E. Hannay, O. Hansen, V.B. Kampenes, A. Karahasanovic, N.-K. Liborg, A.C. Rekdal, A survey of controlled experiments in software engineering, IEEE Trans. Softw. Eng. 31 (9) (2005) 733–753.
[35] S. Barney, M. Khurum, K. Petersen, M. Unterkalmsteiner, R. Jabangwe, Improving students with rubric-based self-assessment and oral feedback, IEEE Trans. Educat. 55 (3) (2012) 319–325.
[36] A. Jonsson, G. Svingby, The use of scoring rubrics: reliability, validity and educational consequences, Educat. Res. Rev. 2 (2) (2007) 130–144.
[37] B. Moskal, K. Miller, L. King, Grading essays in computer ethics: rubrics considered helpful, ACM SIGCSE Bull. 34 (1) (2002) 101–105.
[38] J.W. Wilkerson, J.F. Nunamaker Jr., R. Mercer, Comparing the defect reduction benefits of code inspection and test-driven development, IEEE Trans. Softw. Eng. 38 (3) (2011).
[39] M. Muller, O. Hagner, Experiment about test-first programming, IEE Proc.-Softw. 149 (5) (2002) 131–136.
[40] L. Madeyski, Preliminary analysis of the effects of pair programming and test-driven development on the external code quality, Proc. 2005 Conf. Softw. Eng.: Evol. Emerg. Technol. 130 (2006) 113.
[41] M. Pancur, M. Ciglaric, Impact of test-driven development on productivity, code and tests: a controlled experiment, Inform. Softw. Technol. 53 (6) (2011) 557–573.
[42] A. Gupta, P. Jalote, An experimental evaluation of the effectiveness and efficiency of the test driven development, in: Proceedings of the 1st International Symposium on Empirical Software Engineering and Measurement (ESEM 2007), IEEE Computer Society, Madrid, Spain, 2007, pp. 285–294.
[43] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, C.A. Visaggio, Evaluating advantages of test driven development: a controlled experiment with professionals, in: 5th ACM-IEEE International Symposium on Empirical Software Engineering, September 21, 2006–September 22, 2006, Association for Computing Machinery, 2006, pp. 364–371.
[44] D. Janzen, H. Saiedian, Does test-driven development really improve software design quality?, IEEE Softw 25 (2) (2008) 7784.
[45] J.H. Vu, N. Frojd, C. Shenkel-Therolf, D.S. Janzen, Evaluating test-driven development in an industry-sponsored capstone project, in: 6th International Conference on Information Technology: New Generations, ITNG 2009, April 27, 2009–April 29, 2009, IEEE Computer Society, Las Vegas, NV, United states, 2009, pp. 229–234.
[46] B. George, L.A. Williams, A structured experiment of test-driven development, Inform. Softw. Technol. 46 (2004) 337–342.
[47] D.S. Janzen, H. Saiedian, A leveled examination of test-driven development acceptance, in: 29th International Conference on Software Engineering, ICSE 2007, May 20, 2007–May 26, 2007, IEEE Computer Society, Minneapolis, MN, United states, 2007, pp. 719–722.
[48] C. Desai, D.S. Janzen, J. Clements, Implications of integrating test-driven development into CS1/CS2 curricula, in: 40th ACM Technical Symposium on Computer Science Education, SIGCSE 2009, March 4, 2009–March 7, 2009, Association for Computing Machinery, Chattanooga, TN, United states, 2009, pp. 148–152.
[49] A. Geras, M. Smith, J. Miller, A prototype empirical evaluation of test driven development, in: Proceedings – 10th International Symposium on Software Metrics, METRICS 2004, September 14, 2004–September 16, 2004, IEEE Computer Society, 2004, pp. 405–416.
[50] L. Madeyski, The impact of test-first programming on branch coverage and mutation score indicator of unit tests: an experiment, Inform. Softw. Technol. 52 (2) (2010) 169–184.
[51] D.S. Janzen, H. Saiedian, Test-driven learning in early programming courses, in: 39th ACM Technical Symposium on Computer Science Education, SIGCSE 2008, March 12, 2008–March 15, 2008, Association for Computing Machinery, Portland, OR, United states, 2008, pp. 532–536.
[52] T. Flohr, T. Schneider, An XP experiment with students – setup and problems, in: Proceedings of the 6th International Conference on Product Focused Software Process Improvement, PROFES 2005, 13–15 June 2005, Springer-Verlag, Berlin, Germany, 2005, pp. 474–486.
[53] L. Madeyski, L. Szala, The impact of test-driven development on software development productivity – an empirical study, 2007, pp. 200–211.
[54] D.S. Janzen, H. Saiedian, On the influence of test-driven development on software design, in: Proceedings of the 19th Conference on Software Engineering Education and Training, 2006, IEEE, 2006, pp. 141–148.
[55] M.A. Domino, R.W. Collins, A.R. Hevner, C.F. Cohen, Conflict in collaborative software development, in: Proceedings of the 2003 ACM SIGMIS CPR Conference, April 10, 2003–April 12, 2003, Association for Computing Machinery, Philadelphia, PA, United states, 2003, pp. 44–51.
[56] T. Flohr, T. Schneider, Lessons learned from an xp experiment with students: test-first needs more teachings, in: PROFES, 2006, pp. 305–318.
[57] N. Laranjeiro, M. Vieira, Extending test-driven development for robust web services, in: 2009 2nd International Conference on Dependability, DEPEND 2009, June 18, 2009–June 23, 2009, IEEE Computer Society, Athens, Glyfada, Greece, 2009, pp. 122–127.
[58] L. Huang, M. Holcombe, Empirical assessment of test-first approach, in: 1st Testing: Academic and Industrial Conference – Practice and Research Techniques, TAIC PART 2006, August 29, 2006–August 31, 2006, IEEE Computer Society, Windsor, United kingdom, 2006, pp. 197–200.
[59] T. Dogsa, D. Batic, The effectiveness of test-driven development: an industrial case study, Softw. Qual. J. 19 (4) (2011) 643–661.
[60] S. Bannerman, A. Martin, A multiple comparative study of test-with development product changes and their effects on team speed and product quality, Empir. Softw. Eng. 16 (2) (2011) 177–210.
[61] M. Siniaalto, P. Abrahamsson, A comparative case study on the impact of test-driven development on program design and test coverage, in: 1st International Symposium on Empirical Software Engineering and Measurement, ESEM 2007, September 20, 2007–September 21, 2007, Inst. of Elec. and Elec. Eng. Computer Society, 2007, pp. 275–284.
[62] L.-O. Damm, L. Lundberg, Results from introducing component-level test automation and test-driven development, J. Syst. Softw. 79 (7) (2006) 1001–1014.
[63] O.P.N. Slyngstad, J. Li, R. Conradi, H. Rnneberg, E. Landre, H. Wesenberg, The impact of test driven development on the evolution of a reusable framework of components – an industrial case study, in: 3rd International Conference on Software Engineering Advances, ICSEA 2008, October 26, 2008–October 31, 2008, IEEE Computer Society, Sliema, Malta, 2008, pp. 214–223.
[64] T. Bhat, N. Nagappan, Evaluating the efficacy of test-driven development: industrial case studies, in: ISCE'06 – 5th ACM-IEEE International Symposium on Empirical Software Engineering, September 21, 2006–September 22, 2006, Association for Computing Machinery, 2006, pp. 356–363.
[65] L. Crispin, Driving software quality: how test-driven development impacts software quality, IEEE Softw. 23 (6) (2006) 70–71.
[66] N. Nagappan, E.M. Maximilien, T. Bhat, L. Williams, Realizing quality improvement through test driven development: results and experiences of four industrial teams, Empir. Softw. Eng. 13 (3) (2008) 289–302.
[67] L. Williams, E. Maximilien, M. Vouk, Test-driven development as a defect-reduction practice, in: 14th International Symposium on Software Reliability Engineering, 17–20 November 2003, IEEE Comput. Soc, Los Alamitos, CA, USA, 2003, pp. 34–45.

[68] K. McDaid, A. Rust, B. Bishop, Test-driven development: can it work for spreadsheets? in: Proceedings of the 4th International Workshop on End-user Software Engineering, 2008, p. 2529.
[69] E. Maximilien, L. Williams, Assessing test-driven development at IBM, in: IEEE 25th International Conference on Software Engineering, 3–10 May 2003, IEEE Comput. Soc., Los Alamitos, CA, USA, 2003, pp. 564–569.
[70] M. Ficco, R. Pietrantuono, S. Russo, Bug localization in test-driven development, Adv. Softw. Eng. 2011 (2011) 2.
[71] M. Siniaalto, P. Abrahamsson, Does test-driven development improve the program code? alarming results from a comparative case study, 2007, pp. 143–156.
[72] M.F. Aniche, M.A. Gerosa, Most common mistakes in test-driven development practice: results from an online survey with developers, in: 3rd International Conference on Software Testing, Verification, and Validation Workshops, ICSTW 2010, April 6, 2010–April 10, 2010, IEEE Computer Society, Paris, France, 2010, pp. 469–478.
[73] C. Desai, D. Janzen, K. Savage, A survey of evidence for test-driven development in academia, SIGCSE Bull. 40 (2) (2008) 97–101.
[74] J. Sanchez, L. Williams, E. Maximilien, On the sustained use of a test-driven development practice at IBM, in: AGILE 2007, p. 514.
[75] K. Petersen, Measuring and predicting software productivity: a systematic map and review, Inform. Softw. Technol. 53 (4) (2011) 317–343.
[76] N.E. Fenton, S.L. Pfleeger, Software Metrics: A Rigorous and Practical Approach, Revised, second ed., Course Technology, 1998.
[77] M. Domino, R. Collins, A. Hevner, Controlled experimentation on adaptations of pair programming, Inform. Technol. Manage. 8 (4) (2007) 297–312.
[78] L. Madeyski, Test-Driven Development: An Empirical Evaluation of Agile Practice, Springer, 2010.
[79] L. Zhang, S. Akifuji, K. Kawai, T. Morioka, Comparison between test driven development and waterfall development in a small-scale project, in: Extreme Programming and Agile Processes in Software Engineering, Springer, 2006, pp. 211–212.
[80] S. Rahman, Applying the tbc method in introductory programming courses, in: 37th Annual Frontiers In Education Conference-Global Engineering: Knowledge Without Borders, Opportunities Without Passports, 2007, FIE'07, IEEE, 2007. pp. T1E–20.
[81] S.H. Edwards, Using test-driven development in the classroom: providing students with automatic, concrete feedback on performance, in: Proceedings of the International Conference on Education and Information Systems: Technologies and Applications EISTA, vol. 3, 2003.
[82] B. George, L. Williams, An initial investigation of test driven development in industry, in: Proceedings of the 2003 ACM Symposium on Applied Computing, ACM, 2003, pp. 1135–1139.
[83] S. Xu, T. Li, Evaluation of test-driven development: an academic case study, in: Software Engineering Research, Management and Applications 2009, Springer, 2009, pp. 229–238.
[84] G. Melnik, F. Maurer, A cross-program investigation of students' perceptions of agile methods, in: Proceedings of the 27th International Conference on Software Engineering, 2005, ICSE 2005, IEEE, 2005, pp. 481–488.
[85] P. Abrahamsson, A. Hanhineva, J. Jäälinoja, Improving business agility through technical solutions: a case study on test-driven development in mobile software development, in: Business Agility and Information Technology Diffusion, Springer, 2005, pp. 227–243.
[86] R.A. Ynchausti, Integrating unit testing into a software development teams process, in: Intl. Conf. eXtreme Programming and Flexible Processes in Software Engineering (XP 2001), 2001, pp. 79–83.
[87] S. Yenduri, L.A. Perkins, Impact of using test-driven development: a case study, in: Software Engineering Research and Practice, 2006, pp. 126–129.
[88] R. Kaufmann, D. Janzen, Implications of test-driven development: a pilot study, in: Companion of the 18th Annual ACM SIGPLAN Conference on Object-oriented Programming, Systems, Languages, and Applications, ACM, 2003, pp. 298–299.
[89] C. Gencel, K. Petersen, Worldviews, research methods, and their relationsihpi to validity in empirical software engineering research, in: Proceedings of the International Workshop on Software Measurement (Mensura 2013), 2013.