Factors Limiting Industrial Adoption of Test Driven Development: A Systematic Review

Adnan Causevic, Daniel Sundmark, Sasikumar Punnekkat
Mälardalen University, School of Innovation, Design and Engineering,
Västerås, Sweden
{adnan.causevic, daniel.sundmark, sasikumar.punnekkat}@mdh.se

Abstract — **Test driven development (TDD) is one of the basic practices of agile software development and both academia and practitioners claim that TDD, to a certain extent, improves the quality of the code produced by developers. However, recent results suggest that this practice is not followed to the extent preferred by industry. In order to pinpoint specific obstacles limiting its industrial adoption we have conducted a systematic literature review on empirical studies explicitly focusing on TDD as well as indirectly addressing TDD. Our review has identified seven limiting factors viz., increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain and tool specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code. The results of this study is of special importance to the testing community, since it outlines the direction for further detailed scientific investigations as well as highlights the requirement of guidelines to overcome these limiting factors for successful industrial adoption of TDD.**

Keywords: Test driven developmen; systematic review; agile software development; unit testing; empirical studies.

I. Introduction

Test-driven development (TDD) is an essential part of eXtreme Programming (XP), as proposed by Kent Beck [1]. TDD (referred as test-first programming as well) requires the developers to construct automated unit tests in the form of assertions to define code requirements before writing the code itself. In this process, developers evolve the systems through cycles of test, development and refactoring.

In a recent industrial survey [2], we examined the difference between the preferred and the actual level of usage for a number of contemporary test-related practices. Out of 22 examined practices, TDD gained the highest score of “dissatisfaction” (i.e., the accumulated absolute difference between the preferred and the actual level of usage). Moreover, the preferred level of usage of TDD was significantly higher than the actual level. Hence, the nature of this dissatisfaction could be stated as “Respondents would like to use TDD to a significantly higher extent than they actually do”.

Building upon these previous results, the aim of the current study was to investigate potential factors that are limiting the industrial adoption of TDD. Here, a factor could translate to a method, technique, effect, experience, tool, or similar, that either exists, or missing, or is newly introduced in a particular organisation. The specific research question we address in this paper is:

RQ: *Which factors could potentially limit the industrial adoption of TDD?*

In order to identify such limiting factors, a systematic literature review of empirical studies on TDD was undertaken. Partly based on concerns of an insufficient number of studies due to publication bias [3], the review was not restricted to studies reporting on failure to implement TDD. Instead, we decided to expand the scope of the study and to systematically search for primary empirical studies of TDD, including (1) studies where TDD was the main focus, (2) studies where TDD was one of the investigated practices, and (3) studies where TDD was used in the experimental setting while investigating something else. In case any of the studies reported issue(s) with any specific factors, this was noted. By qualitatively and quantitatively analysing the reported issues on TDD within the selected papers, we have identified a number of limiting factors.

The contributions of this paper are three-fold:

- A qualitative analysis on the effects of a set of factors on TDD, based on reported primary studies
- Identification of a set of factors limiting the adoption of TDD in industry
- Discussions on the implications for research and industry of these factors

The remainder of the paper is organised as follows: Section II provide details of the research method used, Section III presents results and analysis while Section IV discuss the findings of our investigation. The paper is concluded by Section V where conclusion and future work are presented.

II. Research Method

A systematic literature review is an empirical study where a research question or hypothesis is approached by collecting and aggregating evidence from a number of *primary studies* through a systematic search and data extraction process. In this process we followed the guidelines for conducting a systematic literature review proposed by Kitchenham [3].

TABLE I. SEARCHED DATABASES

Source	Search Date
IEEExplore	2010-02-12
ACM Digital Library	2010-02-15
ScienceDirect	2010-02-11
EI Compendex	2010-02-12
SpringerLink	2010-02-12
ISI Web of Science	2010-02-11
Wiley Inter Science Journal Finder	2010-02-16

A. Search Process

The review process started with the development of a review protocol. This protocol described the purpose of the review, defined the research questions as well as preliminary inclusion and exclusion criteria, and provided details on the search string and the databases in which it would be applied.

As for inclusion and exclusion criteria, the search aimed at identifying full-length English language peer-reviewed empirical studies focusing on TDD, including academic and industrial experiments, case studies, surveys and literature reviews. Short papers (in our case, below six pages), tutorials, work-in-progress papers, keynotes, and pure industrial lessons learned were excluded.

Scientific databases in the software engineering field were selected based on the aim of getting a wide and exhaustive coverage of published studies on TDD. The list of selected databases is provided in Table I. Within each of these databases, a search was performed using the following Boolean search string:

"tdd" OR "test driven development" OR "test-driven development" OR "test driven design" OR "test-driven design" OR "test first" OR "test-first" OR "integration driven development" OR "integration-driven development"

The resulting list of primary studies was collected in the EndNote reference management program in order to facilitate the paper exclusion process.

B. Paper Exclusion Process

Following the initial search, which yielded a total of 9.462 papers, exclusion was performed in multiple stages:

1. In the first stage, duplicate papers were removed, and papers were excluded based on formal criteria (e.g., exclusion of short papers) and on title (typically off-topic papers). A total of 509 papers passed this stage.
2. In the second stage, papers were excluded based on abstracts. A total of 150 papers passed this stage.
3. In the third and final exclusion stage, papers were excluded based on full text. In this stage, each paper was read by at least two researchers. To assess the quality and suitability of each study with respect to the review objective, we made use of a review form similar to the screening questions in the quality assessment form used by Dybå and Dyngsøyr in their review of empirical studies of agile development [6]. Specifically, we investigated (1) if the paper was a research paper, (2) if it was an evaluation of TDD, (3) if the research aims were clearly stated, and (4) if the paper had an adequate description of context/setting. In order to pass the stage, a paper had to fulfill both criteria (1) and (2) as well as either (3) or (4). A total of 48 papers passed this stage.

Paper exclusion disagreements were resolved through in-depth discussions between the authors.

TABLE II. EXTRACTED STUDY DETAILS

Extracted study details	
General study information	*Publication type, year, author, etc.*
Study setting	*Academic, industrial or semi-industrial*
Domain of study objective	*Web, business system, embedded system, etc.*
Study type	*Case study, experiment, survey or literature review*
Number of subjects	
Length of study	
Level of experience of subjects	*Novice, medium, experienced*
Focus level of TDD in study	*The main focus of the study is TDD* *One focus of the study is TDD* *TDD is not a focus of the study, but it is used to study something else*

C. Data Extraction Process

In the first step of the data extraction, study details regarding, e.g., study setting and domain, were extracted for all included studies (see Table II). In this step, data extraction was relatively straightforward. However, in the cases where the data of interest was omitted or unclearly stated in the primary study (e.g., failure to mention the level of subjects' previous experience of TDD), the data was omitted from extraction.

The extraction of TDD effects was more complicated. Here, a two-step evolutionary approach was used. In the first step, each selected paper was read by one researcher to identify *explicitly stated effects of TDD* observed in the study. At this stage, there was no discrimination between negative, neutral or positive effects of TDD. This stage of the review also extracted *explicit claims on requirements for a successful adoption of TDD*. The reason for not only extracting negative effects of TDD was that we believe that the resulting partial view would diminish the possibilities of performing a balanced analysis of limiting factors. Once the first step of the data extraction was finished, 10 studies were omitted from further analysis. These studies were either studies that were also reported in other included papers, or contained no explicitly reported effects of TDD.

In the second step, the resulting matrix of TDD effects and primary studies was reviewed for consistency by all authors. The aim was to make sure that the claimed effects had been interpreted similarly in the extraction process.

D. Data Synthesis

Based on the 18 TDD effect areas extracted in the previous step, we defined the limiting factors for the industrial adoption of TDD as effect areas conforming to the following rules:

i. The effect area contained at least two studies with observations of negative effects of or on TDD
ii. The effect area contained more studies with observations of negative effects of or on TDD than it

contained studies with observations of positive effects of or on TDD

iii. Negative effects in the effect area were observed in at least one study performed in an industrial setting.

TABLE III. EMPIRICAL STUDIES OF TDD

Study	Setting	Type	Subjects	Focus level
Abrahamson et. al (2005) [8]	Industrial*	Case Study	Professionals	TDD explicit primary focus
Bhat & Nagappan (2006) [9]	Industrial	Case Study	Professionals	TDD explicit primary focus
Canfora et. al (2006) [10]	Industrial	Experiment	Professionals	TDD explicit primary focus
Canfora et. al (2006) ISESE [11]	Industrial	Experiment	Professionals	TDD explicit primary focus
Cao & Ramesh (2008) [12]	Industrial	Case Study	Professionals	TDD explicit focus, but not main focus
Chien et. al (2008) [13]	Academic	Experiment	Students	TDD explicit focus, but not main focus
Damm & Lundberg (2006) [14]	Industrial	Case Study	Professionals	TDD explicit primary focus
Damm & Lundberg (2007) [15]	Industrial	Case Study	Professionals	TDD explicit primary focus
Domino et. al (2007) [16]	Academic	Experiment	Students	TDD explicit focus, but not main focus
Domino et. al (2003) [17]	Academic	Experiment	Students	TDD not in focus, but used in study setup
Erdogmus et. al (2005) [18]	Academic	Experiment	Students	TDD explicit primary focus
Filho (2006) [19]	Academic	Experiment	Students	TDD not in focus, but used in study setup
Flohr & Schneider (2005) [20]	Academic	Experiment	Students	TDD explicit primary focus
Flohr & Schneider (2006) [21]	Academic	Experiment	Students	TDD explicit primary focus
George & Williams (2003) [22]	Industrial	Experiment	Professionals	TDD explicit primary focus
Geras et. al (2004) [23]	Academic	Experiment	Professionals	TDD explicit primary focus
Gupta & Jalote (2007) [24]	Academic	Experiment	Students	TDD explicit primary focus
Höfer & Philipp (2009) [25]	Academic	Experiment	Mixed	TDD explicit primary focus
Huang & Holcombe (2009) [26]	Academic	Experiment	Students	TDD explicit primary focus
Janzen & Saiedian (2006) [27]	Academic	Experiment	Students	TDD explicit primary focus
Janzen & Saiedian (2008) [28]	Mixed	Experiment/ Case Study	Mixed	TDD explicit primary focus
Janzen et. al (2007) [29]	Academic	Experiment	Professionals	TDD explicit primary focus
Kobayashi et. al (2006) [30]	Industrial	Case Study	Professionals	TDD explicit focus, but not main focus
Kollanus & Isomöttönen (2008) [31]	Academic	Experiment	Students	TDD explicit primary focus
Layman et. al (2006) [32]	Industrial	Case Study	Professionals	TDD explicit focus, but not main focus
LeJeune (2006) [33]	Academic	Case Study	Students	TDD explicit focus, but not main focus
Huang et. al (2007) [34]	Academic	Experiment	Students	TDD explicit focus, but not main focus
Madeyski (2006) [35]	Academic	Experiment	Students	TDD explicit focus, but not main focus
Madeyski (2007) [36]	Academic	Experiment	Students	TDD not in focus, but used in study setup
Madeyski (2008) [37]	Academic	Experiment	Students	TDD not in focus, but used in study setup
Madeyski (2010) [38]	Academic	Experiment	Students	TDD explicit primary focus
Madeyski & Szała (2007) [39]	Industrial	Case Study	Professional	TDD explicit primary focus
Marchenko et. al (2009) [40]	Industrial	Case Study	Professionals	TDD explicit primary focus
Maximilien & Williams (2003) [41]	Industrial	Case Study	Professionals	TDD explicit primary focus
Mišić (2006) [42]	Mixed	Survey	Mixed	TDD explicit focus, but not main focus
Müller & Hagner (2002) [43]	Academic	Experiment	Students	TDD explicit primary focus
Müller & Höfer (2007) [44]	Academic	Experiment	Mixed	TDD explicit primary focus
Nagappan et. al (2008) [45]	Industrial	Case Study	Professionals	TDD explicit primary focus
Salo & Abrahamsson (2007) [46]	Industrial*	Case Study	Professionals	TDD not in focus, but used in study setup
Sanchez et. al (2007) [47]	Industrial	Case Study	Professionals	TDD explicit primary focus
Sfetsos et. al (2006) [48]	Industrial	Survey	Mixed	TDD explicit focus, but not main focus
Sherrell & Robertson (2006) [49]	Academic	Case Study	Students	TDD explicit focus, but not main focus
Siniaalto & Abrahamsson (2007) [50]	Industrial	Case Study	Professionals	TDD explicit primary focus
Siniaalto & Abrahamsson (2008) [51]	Industrial	Case Study	Professionals	TDD explicit primary focus
Slyngstad et. al (2008) [52]	Industrial	Case Study	Professionals	TDD explicit primary focus
Wastnus & Gross (2007) [53]	Industrial	Case Study	Professionals	TDD explicit primary focus
Williams et. al (2003) [54]	Industrial	Case Study	Professionals	TDD explicit primary focus
Vu et. al (2009) [55]	Academic	Experiment	Students	TDD explicit primary focus

*. Close-to-Industry setting as defined in [7]

III. Results and Analysis

This review identified 48 empirical studies concerning TDD. 38 of those included explicit claims on the effects of TDD. This section provides the study details of the larger set of identified studies, as well as an analysis of the limiting factors of TDD, based on the effects of TDD stated in the primary studies.

A. *Empirical Studies of TDD*

An overview of the primary studies on TDD included in our review is given in Table III. Out of the 48 included studies, 25 were experiments, 20 were case studies, 2 were surveys, and one was a mix of a case study and an experiment. 50% of the studies were performed in an academic setting, 46% were studies performed in an industrial setting and 4% were mixed academic/industrial studies. Over half of the included studies (58%) included professional software engineers in the group of study subjects. Most included studies (67%) were studies with TDD as the primary focus of investigation.

Besides the study quality screening used for paper exclusion, we made no further attempt of explicitly evaluating the quality of each included primary study. Even though some additional insights might have been gained by such a quality assessment, we believe that this value would have been limited by the heterogeneity of the included studies.

B. *Reported Effects of and on TDD*

In order to identify limiting factors of industrial adoption of TDD, all included studies were searched for explicit claims on effects **of** TDD (i.e., cause-effect relationships where TDD was the causing factor), as well as explicit claims on effects **on** TDD (i.e., cause-effect relationships where some factor caused an effect on the way TDD was performed). We denote these effect areas of TDD, and Table IV presents the 18 effect areas found in the search. A total count for the number of studies making claims regarding each effect is also given in the table. Again, note that effects were included in this data extraction regardless of whether they were mentioned in a positive, neutral or negative context. As mentioned above, the collection of included primary studies exhibited great heterogeneity. As a consequence, description of TDD effects ranged from purely quantitative data (e.g., ratio scale metrics on code complexity [28][53] or development time [9][22]), to qualitative data based on subjects' responses to open-ended survey questions (e.g., nominal or ordinal scale claims on perception of TDD [33][46]). Consequently, the items in the resulting list of effect areas are not necessarily unique and independent. As an example, effect areas like design, refactoring and complexity are highly related. When doubtful, we have chosen not to group effect areas, as this would result in a less rich information from which to derive the limiting factors of TDD.

TABLE IV. Areas of Effect of TDD

Sl. No.	Description	Count
1	Development time	18
2	Experience/knowledge	4
3	Design	3
4	Refactoring	2
5	Skill in testing	3
6	TDD adherence	8
7	Code quality	18
8	Cost	1
9	Code coverage	8
10	Complexity	7
11	Time for feedback	5
12	Domain and tool specific issues	10
13	Code size	3
14	Perceptions	15
15	Communication & (customer) collaboration	1
16	Legacy code	2
17	Defect reproduction	1
18	Documentation	1

A more detailed view of the areas of effect of TDD with respect to the primary studies is provided in Table V.

C. *Factors Limiting Industrial Adoption of TDD*

Based on the effect areas presented in the previous section, we identify seven limiting factors (LF1-LF7) for industrial adoption of TDD. An overview of these factors is given in Table VI. We now describe each of these limiting factors in detail together with the observations from the primary studies as well as providing motivations for their inclusions.

1) ***LF1: Increased development time***

Description: By *development time,* we refer to the time required to implement a given set of requirements. Time required for development of software product is relatively easy to measure. It is however a matter of discussion whether the time for corrective re-work (e.g., based on failure reports from later testing stages) should be included in the development time or not.

Observations: Nine included primary studies in the review reported negative experience with respect to the time for development. Six were industrial studies with professionals (five case studies and one experiment) and three were academic studies with students (two experiments and one case study). Five studies did report positive effect on development time when using TDD, but this was mostly when the overall project time was captured.

Discussion: Development time could be considered a business-critical factor for adopting new practices within an organisation. Depending on the maturity of the organization, an up-front loss (in this case, increased development time) might overshadow a long-term gain (e.g., decreased overall project time, or increased product quality – both of which were reported in many of our included studies). Hence, internal organizational pressure might risk the proper usage of TDD.

2) LF2: Insufficient TDD experience/knowledge

Description: By TDD experience/knowledge, we refer to the degree of practical experience, as a developer or similar. or theoretical insight in TDD.

Observations: Two industrial case studies with professional developers attributed problems of implementing TDD to lack of TDD education/experience. Moreover, two other studies report significant differences in the way of applying TDD between experienced TDD developers and novice TDD developers.

Discussion: When observing collected data from the included primary studies, we noticed that participants in the experiments (either students or professionals) were mostly provided with some training or tutorial on how to perform TDD. In several cases [46], the knowledge improved as participants would progress with the experiment. We expect that lack of knowledge or experience with TDD could create problems in its adoption.

3) LF3: Insufficient design

Description: *Design,* in this context, refers to the activity of structuring (or re-structuring) the system or software under development or in evolution in order to avoid architectural problems, and to improve architectural quality. Detailed up-front software design is a common practice of plan-driven development methodologies. TDD emphases on a small amount of up-front design, and frequent refactoring to keep the architecture from erosion.

Observations: Three primary studies reported architectural problems when using TDD. These were two academic experiments with students and one industrial case study with professionals.

Discussion: There is no massive empirical support that the lack of design should be considered as a limiting factor for industrial adoption of TDD. However, there are a handful of studies reporting problems regarding lack of design in TDD, particularly in the development of larger, more complex systems. Moreover, the lack of upfront design has been one of the main criticisms of TDD since its introduction and even if the evidence supporting this criticism is sparse, so is the evidence contradicting it [57].

4) LF4: Insufficient developer testing skills

Description: By *developer testing skill*, we refer to the developer's ability to write efficient and effective automated test cases.

Observations: Two of the included primary studies report negative experiences with developers' testing. One study is an academic experiment with student subjects, where it is reported that students expressed difficulties to come up with good test cases. The other study is an industrial case study with mixed professional and student subjects where lack of developer testing skills was stated as a limiting factor.

Discussion: Since TDD is a design technique where the developer undertakes development by first creating test cases and then writes code that makes the test cases pass, it relies on the ability of the developer to produce sufficiently good test cases. Additionally, Geras [27] reports on the risk it brings to adopt TDD without having adequate testing skills and knowledge. We find it interesting that there are no explicit investigations of the quality of test cases produced by developers in TDD.

5) LF5: Insufficient adherence to the TDD protocol

Description: By *adherence to the TDD protocol,* we refer to the degree to which the steps of the TDD practice are followed. For example, are test cases always created and exercised to failure before the corresponding code is written? TDD is a defined practice with fairly exact guidelines on how it should be executed.

Observations: Related to this limiting factor, there are two types of observations that are of relevance. First, three industrial case studies, with professionals as subjects, report negative experiences with developer adherence to the TDD protocol. Reasons for abandoning the TDD protocol included time pressure, lack of discipline, and shortage of perceived benefits. Second, two additional industrial case studies, with professionals as subjects, reported correlations between low TDD adherence and low quality. It is noteworthy that these observations were made in organizations where TDD was the preferred development method.

Discussion: The combined view of the five above mentioned industrial case studies motivate the inclusion of the lack of TDD adherence as a limiting factor. Basically, the studies state that (1) it is important to adhere to the TDD protocol, and (2) developers do stray from the protocol in several situations. It is however far from certain that there is a clean-cut cause-effect relationship between low TDD adherence and low quality. Not unlikely, confounding factors (e.g., tight development deadlines) might lead to both low TDD adherence and poor quality.

6) LF6: Domain- and tool-specific limitations

Description: By *domain- and tool-specific limitations*, we refer to technical problems in implementing TDD (e.g., difficulties in performing automated testing of GUIs). Generally, the TDD practice requires some tool support in the form of automation framework for test execution.

Observations: Nine studies reported negative experiences with respect to domain and tool-specific issues. Five of them were industrial case studies with professionals as subjects, one was an industrial survey with both student and professional respondents and three were academic experiments with student subjects. The single most reported issue is the problem of automatically testing GUI applications, but also networked applications seem to be problematic in terms of automated testing.

Discussion: Proper tool support for test automation is vital for the successful adoption of TDD. With the wide variety of studies reporting domain- and tool-specific issues as a limiting factor in the adoption of TDD, the factor would be difficult to ignore.

TABLE V. MAPPING BETWEEN EFFECT OBSERVATIONS AND PRIMARY STUDIES

Study \ Effect	Development time	Experience/knowledge	Design	Refactoring	Skill in testing	TDD adherence	Code quality	Cost	Code coverage	Complexity	Time for feedback	Domain and tool specific issues	Code size	Perceptions	Communication & customer collaboration	Legacy code	Defect reproduction	Documentation
Abrahamson et. al (2005) [8]		–				–						–						
Bhat & Nagappan (2006) [9]	–						+											
Cao & Ramesh (2008) [12]	–					–					+							
Damm & Lundberg (2007) [15]												–						
Domino et. al (2007) [16]						!	=											
Erdogmus et. al (2005) [18]							=											
Filho (2006) [19]							+											
Flohr & Schneider (2005) [20]												–						
Flohr & Schneider (2006) [21]	+								=					!				
George & Williams (2003) [22]	–						+		+					+				
Geras et. al (2004) [23]	!				!		!											
Gupta & Jalote (2007) [24]	+		–															
Höfer & Philipp (2009) [25]		!		!		!												
Huang & Holcombe (2009) [26]	–						–											
Janzen & Saiedian (2006) [27]	+												–	+				
Janzen & Saiedian (2008) [28]										+			+	!				
Janzen et. al (2007) [29]									+					+				
Kobayashi et. al (2006) [30]	+						+	+										
Kollanus & Isomöttönen (2008) [31]			–		–							–		!				
Layman et. al (2006) [32]						–						–				–		
LeJeune (2006) [33]	–						+							+				
Madeyski (2010) [38]									=					–				
Marchenko et. al (2009) [40]	–		–				+			+		!		!		–	–	+
Maximilien & Williams (2003) [41]	–/=	!					+				+			+				
Mišić (2006) [42]						+												
Müller & Hagner (2002) [43]	–/=																	
Müller & Höfer (2007) [44]						!			!		!							
Nagappan et. al (2008) [45]	–					!	+											
Salo & Abrahamsson (2007) [46]		–										–		+				
Sanchez et. al (2007) [47]	–						+			+								
Sfetsos et. al (2006) [48]					–						+	–			!			
Sherrell & Robertson (2006) [49]														–				
Siniaalto & Abrahamsson (2007) [50]									+	–								
Siniaalto & Abrahamsson (2008) [51]							!			!								
Slyngstad et. al (2008) [52]				+			+											
Wastnus & Gross (2007) [53]	+						+		+	+	+	–		+/–				
Williams et. al (2003) [54]	=						+							+				
Vu et. al (2009) [55]	–						+		–	+/!		–	!	+				

Table legend:

+	positive mentioning of a particular effect	=	no effect was reported for a particular effect
–	negative mentioning of a particular effect	!	effect was mentioned as an important observation

TABLE VI. LIMITING FACTORS FOR TDD ADOPTION

Label	Description
LF1	Development time
LF2	Experience/knowledge
LF3	Design
LF4	Skill in testing
LF5	TDD adherence
LF6	Domain and tool specific issues
LF7	Legacy code

7) *LF7: Legacy code*

Description: By *legacy code,* we refer to the existing codebase in a development organization. Legacy code often represent decades of development efforts and investments, and serve as a backbone both in existing and future products.

Observation: Two industrial case studies with professionals as subjects report problems with handling the legacy codebase in an adoption of TDD. Particularly, the automated regression test suites on unit level (which are natural consequences of long-term TDD development), are often missing for legacy code.

Discussion: TDD, in its original form, does not discuss how to handle legacy code. Instead, the method seems to assume that all code is developed from scratch, using TDD as the development method. As this is seldom the case in large development organization, adoption of TDD might be problematic. A lack of automated regression suites for legacy code hampers the flexibility provided by the quick feedback on changes provided by the regression suites, and may leave developers more anxious about how new changes may unexpectedly affect existing code.

IV. DISCUSSION

In this section we are discussing threats to validity of our research as well as implications of our results on research and industry.

A. *Threats to Validity*

Typically, four types of validity are discussed in empirical research (i.e., *construct validity, internal validity external validity* and *reliability*) [56]. Below, the threats to these validities in our study are discussed.

Construct Validity refers to the correctness in the mapping between the theoretical constructs that are to be investigated, and the actual observations of the study. In a systematic review, the validity of the study constructs is inherited from the construct validity in the included primary studies. In our case, this validity threat concerned both the actual treatment of the primary studies (i.e., TDD) and the effect on study outcomes (e.g, quality or development time).

First, in order to measure the effects of TDD in an empirical study, one must be sure that TDD is actually used within the study. This problem was handled differently in different studies. Some studies merely assumed that TDD was used by the set of subjects as instructed [19], some studied used manual supervision [8], and some studies used elaborate tools to ensure TDD adherence [23]. Second, with respect to outcome measures (i.e., the effects of TDD), the construct validity is different for different constructs. Most metrics for, e.g., complexity are formally defined and measured, and are hence not subjects to threats to construct validity. However, constructs like design quality and developer skill are subject to interpretation in each primary study. In the review, we sought to mitigate this threat by performing the data extraction in two phases, with the second phase focusing on a conformance in the interpretation of primary study constructs between the authors.

Internal Validity concerns the proper analysis of data. Given the heterogeneity of the included primary studies in the review, internal validity is a subject of concern, particularly in statistical analysis of the extracted data. However, we draw no generalized statistical conclusions regarding the effects of TDD. Rather, our contribution is a set of directions for future research and industrial guidelines, based on a qualitative analysis of the extracted data.

External Validity relates to the possibility to generalize the study results outside its scope of investigation. The variety of study setting, type and domain serves to limit the external validity threat of the review, particularly in the cases where limiting factors are found across several studies in different domains. Also, by collecting study details, we had the possibility to differentiate results based on particular study details.

Reliability concerns the degree of certainty with which a replication of this study, e.g., by a different set of researchers, would yield the same study outcome. As the search strategy, as well as the inclusion and exclusion criteria, is explicitly given in this study, the main reliability threat concerns the analysis resulting in the aggregation from reported effects of TDD to limiting factors. Particularly effects with low construct validity may be interpreted differently in replicated reviews. Hence, we have sought to describe the research process, including the data analysis, in a transparent manner.

B. *Implications for Research*

Test driven development was and still is under constant investigation of researchers who are providing evidence of claimed benefits which this practice can bring to a software development team. These benefits can be also seen in our mapping table (Table V) between effect observations and primary studies. Most noticeable positive effect is a code quality improvement which is one of the reasons why TDD is gaining interest. Also we can see that primary studies are reporting a positive perception of participants towards TDD. This is something our previous study [2] also revealed.

Having those two benefits empirically addressed (quality improvement and positive perception of practice) we propose that the next empirical evaluation on TDD should include a direct investigation on the impact of the limiting factors we presented in Section III.

Next studies could focus on a limiting factor of development time (LF1) together with the lack of TDD experience factor (LF2) to investigate if the actual learning curve could be generating additional time for the development. However, it is important for researchers to clearly state if increased development time was reported during unit development or it is reflected on overall project. This is something we had difficulties extracting from primary studies.

By providing more complex algorithms or working in a different domain of investigation (safety-critical systems on embedded device for example) researchers could investigate how lack of up-front design (LF3) influence adoption of TDD. Even in a student experiment setup, Kollanus [31] noted that slightly complex application required more of up-front design.

TDD is a development technique which requires from developers to write test cases. We noticed that primary studies are not directly investigating how these test cases are designed and whether designing test cases for TDD is different from how experienced testers are performing it. By investigating LF4 with independent teams of experienced testers and developers with the focus on efficiency and quality of test design, researchers could gain insights on issues such as 1) whether lack of quality in tests is limiting adoption of TDD, and 2) the right level of testing education required for developers to perform TDD.

Regarding LF5, TDD adherence needs to be further evaluated. A first step would be to make sure that all studies examining the effects of TDD also has some means of measuring TDD adherence in the experimental setting. Particularly in industrial case studies, it would be valuable to investigate TDD adherence over time, with variations in, e.g., TDD experience, workload and type of development task. Such observations should be correlated with resulting measurements on quality and development time.

Regarding the more technical limiting factors (i.e., the domain- and tool-specific issues (LF6), and the lack of automated regression suites for legacy code (LF7)), it is our belief that research could contribute with improved methods and techniques for different aspects of test automation, including automated test case generation, test case execution and test case evaluation.

C. *Implications for Industry*

This review identifies a set of potentially limiting factors of industrial adoption of TDD, based on aggregated observations from TDD usage in various different settings. Consequently, a set of industrial guidelines can be derived from the study results.

First, the limiting factors that are controllable and specific to the adopting organization should be taken into account prior to TDD adoption. Specifically, proper training on TDD practice (relating to LF2) and test case design (relating to LF4) should be provided before the adoption. Additionally, strategic recruitments of experienced TDD developers, who might serve as TDD mentors could limit the problems related to lack of TDD experience. Moreover, TDD adoption should be considered in the light of the organizational domain (relating to LF6). For example, are the developed systems heavy on graphical user interaction? Is there proper tool support in the existing development infrastructure for the level of test automation required by TDD? In addition, if the adopting organization includes a significant legacy codebase lacking automated regression test suites (relating to LF7), there should be a strategy of how to handle this that does not collide with the TDD development protocol.

Second, the limiting factors that are general and TDD-inherent should be considered and monitored in the adopting organization (with respect to that organization's motives for adopting TDD). Specifically, it should be considered whether a small increase in (unit-level) development time, if observed, would be acceptable to reach other expected benefits of TDD (relating to LF1). Moreover, it would be advisable to track the architectural quality (both based on metrics of architectural quality attributes and developer perception) to ensure that the lack of upfront design does not lead to architectural erosion (relating to LF3). In addition, although the enforcing TDD protocol upon the developers might not be a good idea, it might still be advisable to keep track of the effects on, e.g., quality, in situations where the TDD protocol is not followed (relating to LF5).

V. CONCLUSION

In this paper we present our analysis results from a systematic review of the empirical studies reported in the literature on the effects of various factors on Test Driven Development. We identified 18 effects, out of which 7 were deemed as limiting factors on the industrial adoption of TDD. These factors were increased development time, insufficient TDD experience/knowledge, lack of upfront design, domain and tool specific issues, lack of developer skill in writing test cases, insufficient adherence to TDD protocol, and legacy code. We also provided reasons for their inclusion as well as discussions on the various implications of these factors. We also outlined the future research and industrial challenges in the light of these findings.

From the perspective of the software testing community, this study throws open the following interesting research challenges to be addressed:

- What is the optimum level of testing knowledge essential for a TDD developer to be efficient?
- Are there any fundamental changes warranted by the TDD approach in the test design techniques?
- How to integrate the TDD perspective of testing and the testers' perspective of traditional development to provide a unified theory for bringing synergy between development and testing efforts in a more productive manner?
- In what new roles [58] and how testers could contribute with their knowledge and experience in the adoption of TDD process within an organisation?

ACKNOWLEDGMENTS

This work was supported by MRTC (Mälardalen Real-Time Research Centre) and the SWELL (Swedish software Verification & Validation ExceLLence) research school. Authors would like to express their gratitude to Dr. Stig Larsson for valuable comments to the review protocol.

REFERENCES

[1] K. Beck, “Extreme programming eXplained: embrace change”, Addison-Wesley, Reading (2000)

[2] A. Causevic, D. Sundmark, S. Punnekkat, “An Industrial Survey on Contemporary Aspects of Software Testing”, Third International Conference on Software Testing, Verification and Validation (ICST), 2010

[3] B. A. Kitchenham, S. Charters, “Guidelines for performing Systematic Literature Reviews in Software Engineering”, Version 2.3, Keele University, EBSE Technical Report, EBSE-2007-01, 2007

[4] T. Dybå, T. Dingsøyr, “Empirical Studies of Agile Software Development: A Systematic Review”, Information and Software Technology, 2008

[5] P. Sfetsos, I. Stamelos, “Empirical Studies on Quality in Agile Practices: A Systematic Literature Review”, Quality of Information and Communications Technology (QUATIC), 2010

[6] J. E. Hannay, T. Dybå, E. Arisholm, D. I. Sjøberg, ”The effectiveness of pair programming: A meta-analysis”, Inf. Softw. Technol. 51, 2009

[7] O. Salo, P. Abrahamsson, ”Empirical Evaluation of Agile Software Development: A Controlled Case Study Approach”, Product Focused Software Process Improvement, 2004

[8] P. Abrahamsson, A. Hanhineva, J. Jäälinoja, “Improving business agility through technical solutions: a case study on test-driven development in mobile software development”, IFIP International Federation for Information Processing, 2005, vol. 180. Springer. pp. 1-17.

[9] T. Bhat, N. Nagappan, “Evaluating the efficacy of test-driven development: industrial case studies”, International Symposium on Empirical Software Engineering, 2006

[10] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, C. A. Visaggio, “Productivity of Test Driven Development: A Controlled Experiment with Professionals”, Product-Focused Software Process Improvement (PROFES), 2006

[11] G. Canfora, A. Cimitile, F. Garcia, M. Piattini, C. A. Visaggio, “Evaluating advantages of test driven development: A controlled experiment with professionals”, International Symposium on Empirical Software Engineering, 2006

[12] L. Cao, B. Ramesh, “Agile Requirements Engineering Practices: An Empirical Study”, Software, IEEE , vol.25, 2008

[13] L. R. Chien, D. J. Buehrer, C. Y. Yang, C. M. Chen, “An Evaluation of TDD Training Methods in a Programming Curriculum”, International Symposium on It in Medicine and Education, 2008

[14] L.-O. Damm, L. Lundberg, “Results from introducing component-level test automation and Test-Driven Development”, Journal of Systems and Software, 2006

[15] L.-O. Damm, L. Lundberg, “Quality impact of introducing component-level test automation and test-driven development”, Proceedings of the 14^{th} European Conference on Systems & Software Process Improvement and Innovation (EuroSPI), 2007, Lecture Notes in Computer Science, vol. 4764. Springer. pp. 187-199.

[16] M. A. Domino, R. W. Collins, A. R. Hevner, “Controlled experimentation on adaptations of pair programming”, Inf. Technol. and Management 8, 2007

[17] M. A. Domino, R. W. Collins, A. R. Hevner, C. F. Cohen, “Conflict in collaborative software development”, SIGMIS, 2003

[18] H. Erdogmus, M. Morisio, M. Torchiano, “On the Effectiveness of the Test-First Approach to Programming”, IEEE Trans. Software Engineering 31, 2005

[19] W.P.P. Filho, “Quality gates in use-case driven development”, ICSE Workshop on Software Quality, 2006

[20] T. Flohr, T. Schneider, “An XP Experiment with Students - Setup and Problems”, Profes, 2005

[21] T. Flohr, T. Schneider, “Lessons learned from an XP Experiment with Students: Test-First needs more teachings”, Profes 2006, Lecture Notes in Computer Science, Volume 4034/2006, 305-318.

[22] B. George, L. Williams, “A Structured Experiment of Test-Driven Development”, Information and Software Technology 46(5), pp. 337--342, 2003

[23] A. Geras, M. Smith, J. Miller, ”A Prototype Empirical Evaluation of Test Driven Development”, METRICS, 2004

[24] A. Gupta, P. Jalote, “An Experimental Evaluation of the Effectiveness and Efficiency of the Test Driven Development”, First international Symposium on Empirical Software Engineering and Measurement, 2007

[25] A. Höfer, M. Philipp, “An Empirical Study on the TDD Conformance of Novice and Expert Pair Programmers”, XP, 2009

[26] L. Huang, M. Holcombe, “Empirical investigation towards the effectiveness of Test First programming”, Inf. Softw. Technol. 51, 2009

[27] D. S. Janzen, H. Saiedian, “On the Influence of Test-Driven Development on Software Design”, Conference on Software Engineering Education & Training, 2006

[28] D. Janzen, H. Saiedian, “Does Test-Driven Development Really Improve Software Design Quality?”, IEEE Software, 25, 2008

[29] D. S. Janzen, C. S. Turner, H. Saiedian, “Empirical Software Engineering in Industry Short Courses”, Conference on Software Engineering Education & Training, CSEET, 2007

[30] O. Kobayashi, M. Kawabata, M. Sakai, E. Parkinson, ”Analysis of the interaction between practices for introducing XP effectively”, 28th international Conference on Software Engineering, ICSE, 2006

[31] S. Kollanus, V. Isomöttönen, ”Understanding TDD in academic environment: experiences from two experiments”, 8th international Conference on Computing Education Research, 2008

[32] L. Layman, L. Williams, L. Cunningham, “Motivations and measurements in an agile case study”, Journal Syst. Archit. 52, 2006

[33] N. F. LeJeune, “Teaching software engineering practices with Extreme Programming”, Comput. Small Coll. 21, 2006

[34] L. Huang, C. Thomson, M. Holcombe, “How good are your testers? An assessment of testing ability”, TAIC-PART 2007

[35] L. Madeyski, “The impact of pair programming and test-driven development on package dependencies in object-oriented design - an experiment”, Lecture Notes in Computer Science, vol. 4034. 2006

[36] L. Madeyski, “On the effects of pair programming on thoroughness and fault-finding effectiveness of unit tests”, Product-Focused Software Process Improvement v.4589, 2007

[37] L. Madeyski, “Impact of pair programming on thoroughness and fault detection effectiveness of unit test suites”, Softw. Process 13, 2008

[38] L. Madeyski, “The impact of Test-First programming on branch coverage and mutation score indicator of unit tests: An experiment”, Inf. Softw. Technol. 52, 2010

[39] L. Madeyski, Ł. Szała, “The impact of test-driven development on software development productivity - an empirical study”, Lecture Notes in Computer Science, vol. 4764. 2007

[40] A. Marchenko, P. Abrahamsson, T. Ihme, “Long-Term Effects of Test-Driven Development A Case Study”, XP 2009

[41] E. M. Maximilien, L. Williams, “Assessing test-driven development at IBM”, 25th international Conference on Software Engineering, ICSE 2003

[42] V. B. Mišić, “Perceptions of extreme programming: an exploratory study”, SIGSOFT Software Engineering Notes 31, 2006

[43] M. M. Müller, O. Hagner, “Experiment about test-first programming”, IEE Procedings-Software. v149, 2002

[44] M. M. Müller, A. Höfer, ”The effect of experience on the test-driven development process”, Empirical Software Engineering. 12, 2007

[45] N. Nagappan, E. M. Maximilien, T. Bhat, L. Williams, “Realizing quality improvement through test driven development: results and experiences of four industrial teams”, Empirical Software Engineering 13, 2008

[46] O. Salo, P. Abrahamsson, “An iterative improvement process for agile software development”, Software Process Improvement and Practice. v12., 2007

[47] J. C. Sanchez, L. Williams, E. M. Maximilien, “On the Sustained Use of a Test-Driven Development Practice at IBM”, AGILE, 2007

[48] P. Sfetsos, L. Angelis, I. Stamelos, ”Investigating the extreme programming system – An empirical study”, Empirical Software Engineering. 11, 2006

[49] L. B. Sherrell, J. J. Robertson, “Pair programming and agile software development: experiences in a college setting”, Journal Comput. Small Coll. 22, 2006

[50] M. Siniaalto, P. Abrahamsson, “A Comparative Case Study on the Impact of Test-Driven Development on Program Design and Test Coverage”, Empirical Software Engineering and Measurement, ESEM, 2007

[51] M. Siniaalto, P. Abrahamsson, “Does Test-Driven Development Improve the Program Code? Alarming Results from a Comparative Case Study”, CEE-SET, 2008

[52] Slyngstad et. al “The Impact of Test Driven Development on the Evolution of a Reusable Framework of Components – An Industrial Case Study”, Third international Conference on Software Engineering Advances, ICSEA, 2008

[53] H. Wastnus, H. G. Gross, “Evaluation of test-driven development - An industrial case study”, Second International Conference on Evaluation of Novel Approaches to Software Engineering, ENASE 2007

[54] L. Williams, E. M. Maximilien, M. Vouk, “Test-Driven Development as a Defect-Reduction Practice”, 14th international Symposium on Software Reliability Engineering, ISSRE, 2003

[55] J. H. Vu, N. Frojd, C. Shenkel-Therolf, D. S. Janzen, “Evaluating Test-Driven Development in an Industry-Sponsored Capstone Project”, Sixth international Conference on information Technology: New Generations, ITNG, 2009

[56] C. Wohlin, P. Runeson, M. Höst, M. C. Ohlsson, B. Regnell, A. Wesslén, “Experimentation in Software Engineering: an Introduction”, Kluwer Academic Publishers, 2000

[57] H. Pei-Breivold, D. Sundmark, P. Wallin, S. Larsson, “What Does Research Say About Agile and Architecture?”, International Conference on Software Engineering Advances, ICSEA, 2010

[58] A. Causevic, A. Sajeev, S. Punnekkat, “Redefining the role of testers in organisational transition to agile methodologies”, International Conference on Software, Services & Semantic Technologies (S3T), 2009