Information and Software Technology 68 (2015) 82–97

Contents lists available at ScienceDirect

Information and Software Technology

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

Towards an operationalization of test-driven development skills: An industrial empirical study

Davide Fucci [a,*], Burak Turhan [a], Natalia Juristo [a,b], Oscar Dieste [a], Ayse Tosun-Misirli [c], Markku Oivo [a]

[a] M-Group, University of Oulu, 90014 Oulu, Finland
[b] Escuela Tecnica Superior de Ingenieros Informaticos, Universidad Politecnica de Madrid, Campus Montegancedo, Calle ciruelos, s/n, 28660 Boadilla del Monte, Madrid, Spain
[c] Faculty of Computer and Informatics, Istanbul Technical University, 34469 Maslak Istanbul, Turkey

ARTICLE INFO

Article history:
Received 11 March 2015
Revised 22 August 2015
Accepted 23 August 2015
Available online 8 September 2015

Keywords:
Test-driven development
Process conformance
Software quality
Developers' productivity

ABSTRACT

Context: The majority of the empirical studies on Test-driven development (TDD) are concerned with verifying or refuting the effectiveness of the technique over a traditional approach, and they tend to neglect whether the subjects possess the necessary skills to apply TDD, though they argue such skills are necessary.
Objective: We evaluate a set of minimal, a priori and in process skills necessary to apply TDD. We determine whether variations in external quality (i.e., number of defects) and productivity (i.e., number of features implemented) can be associated with different clusters of the TDD skills' set.
Method: We executed a quasi-experiment involving 30 practitioners from industry. We first grouped the participants according to their TDD skills' set (consisting of a priori experience on programming and testing as well as in-process TDD conformance) into three levels (Low-Medium-High) using k-means clustering. We then applied ANOVA to compare the clusters in terms of external quality and productivity, and conducted post-hoc pairwise analysis.
Results: We did not observe a statistically significant difference between the clusters either for external software quality ($F(2, 27 = 1.44,\ p = .260$), or productivity ($F(2, 27) = 3.02,\ p = .065$). However, the analysis of the effect sizes and their confidence intervals shows that the TDD skills' set is a factor that could account for up to 28% of the external quality, and 38% for productivity.
Conclusion: We have reason to conclude that focusing on the improvement of TDD skills' set investigated in this study could benefit software developers in improving their baseline productivity and the external quality of the code they produce. However, replications are needed to overcome the issues related with the statistical power of this study. We suggest practical insights for future work to investigate the phenomenon further.

1. Introduction

Test-driven development (TDD) is a software development technique in which the development is guided by writing unit tests. It was popularized in the late 1990s as part of Extreme Programming [1]. A developer using TDD follows four steps:

1. Write a unit test for the functionality she wants to add.
2. Run the unit test to make sure it fails.
3. Write only enough production code to make the test to pass.
4. Refactor both production and test code, and re-run the tests.

TDD is claimed to yield better results than traditional approaches to software development (e.g., when unit tests are written after the intended functionality is considered *completed* by the development team) in terms of developers' productivity, external quality (e.g., reduced number of defects), maintainability, and extensibility [2,3]. However, empirical investigations of the effects of TDD are contrasting [4,5], arguing that the results are influenced by several variables (e.g., academic vs. industrial settings), including the *skills of developers*.

Literature reviews on TDD conclude that the application of the technique—and subsequently the manifestation of its postulated benefits—requires some skills [5,6]; however, these studies do not indicate what these skills are. We started our investigation on skills with students in a previous study [7]. In that context, we looked at their pre-existing knowledge regarding two practical skills: proficiency with programming language and unit testing (UT). When the subjects tackled a small programming task using TDD, we found that such skills had little impact on their productivity—defined as the

* Corresponding author. Tel.: +358445013562.
E-mail address: davide.fucci@oulu.fi (D. Fucci).

http://dx.doi.org/10.1016/j.infsof.2015.08.004
0950-5849/

output (e.g., parts of the task completed) per unit of effort (e.g., time to complete the task). No significant relationship was observed regarding the quality of the software they produced—e.g., the defects found in the parts of the task which were completed by the subjects. In the same study, we acknowledged that other skills must be present in order for TDD developers to achieve the benefits advocated by TDD supporters.

With these motivations based on existing literature and our previous work, we incorporate in this study another practical skill, which we call *TDD process conformance*, along with programming language and unit test skills. *TDD process conformance* represents the ability of a developer to follow the TDD cycle. Together, these three skills represent our *TDD skill set*. Further, we used a more realistic task to overcome the limitations of small programming tasks, and recruited professional developers for the study. Consequently, the research goal of this work is the following:

> Understanding the effect of the developers' TDD skills on external quality and productivity

In our previous studies [7–9] we have investigated the role that each skill plays *individually* with student subjects working on toy tasks. We now focus on the impact the skills have, when taken *together*, on the outcomes of interest, by performing a quasi-experiment involving 43 professional software developers (30 after mortality) without prior working experience in TDD. The developers were trained during a week-long workshop and then asked to implement new features of a legacy system using TDD. Finally, we evaluated the composite effect of their skills on their performance in terms of external quality and productivity. Hence, we contribute to the existing knowledge by:

- Empirically investigating an anecdotal claim: that is, TDD requires skills to manifest benefits, with professional developers.
- Building a model for quality and productivity that takes into account a set of practical skills (Section 3)
- Providing initial empirical evidence that further investigation of the proposed *TDD skill set* are worth pursuing (Section 5)

The strong points of our study lie in the settings (Section 4) in which it was conducted. In particular, we:

- Analyze data collected from professional software developers.
- Utilize a near real-world, brown-field task, rather than a toy, green-field, task (see Section 4.2 and Appendix B).
- Quantify process conformance analytically, rather than relying on self reports.

The rest of the paper is organized as follows. In Section 2 we present the existing literature related to our research, in Section 3 we define the TDD skill set used in our study. Section 4 explains the details of our empirical study design. Sections 5 and 6 report the results and associated discussions. We address the threats to the validity of our study in Section 7. We conclude the paper in Section 8.

2. Related work

Test-driven development has been the subject of several secondary studies. The systematic literature review by Turhan et al. [5]—covering 32 empirical studies—found positive effects on external quality, whereas the productivity results were inconclusive, when TDD was used across different settings. The meta-analysis by Rafique and Misic [4] is of interest when looking at how experience works with the postulated TDD effects. The work covers 10 years of TDD publications, from 2000 to 2011, in 25 selected primary studies. The authors focused part of their analysis on comparing studies whose subjects had different kinds of experience, i.e., academic vs. industrial. The results show improvement for professionals in terms of external quality, but a deterioration of productivity compared to student subjects.

In a recent systematic literature review, Munir et al. [10] classified the primary studies according to relevance and rigor. In particular, relevant studies, i.e., studies dealing with realistic settings that have applicability in an industrial context, show that TDD benefits professional developers in terms of external quality at the expense of productivity. Nevertheless, the authors suggest that there is a lack of industry experiments dealing with real-world systems and long-term studies.

Based on the big picture provided by the systematic literature reviews, it appears that the goal of TDD research (including the secondary studies) is to gather evidence about TDD beneficial effects over a traditional approach to software development, like test-last development. We acknowledge the importance of such research effort, but we also note that the majority of the empirical work pays insufficient attention to whether the subjects possess the necessary skills, and apply such skills in a test-driven fashion. Moreover, prior research defines experience in terms of subject roles, e.g., students vs. professionals.

Latorre [11] studied the effects of the application of TDD by a pool of professional software developers (i.e., having skills with Java programming, and UT in JUnit but not TDD) to a real-world, although simple, software system, over a one-month period. The author shows that the developers were able to apply TDD correctly after a short practice and retain such knowledge later in their daily work. When the subjects were considered according to their seniority (i.e., junior, intermediate, and senior), the results show that the ability to readily apply TDD initially depends on experience. In fact, senior developers were able to achieve a high level of conformance to the process after few iterations, while intermediate and juniors needed more time, after which, all the subjects reached a plateau level between 80% and 90%. On the other hand, experience had an impact on productivity. Only the most expert subjects were able to keep the productivity at the level of a traditional development approach (the initial part of the system was developed without employing TDD), while the less experienced ones lagged behind due to the problems they encountered with refactoring and design decisions. Nonetheless, all the subjects delivered a correct and functioning version of the system. Therefore—although not explicitly mentioned by the author—external quality does not seem to be affected by the subjects' experience or level of conformance. The author advises that similar studies should be repeated by taking into account different levels of experience with the programming language, UT, and tools, as well as real-world application, since such factors might affect the adoption of TDD.

Another study inspecting the role of experience and process conformance in TDD settings is the controlled experiment by Müller and Höfer [12], in which experienced and novice developers were compared. The experts in this case also had previous knowledge of Java (average 6.4 years), JUnit (average 4.3 years), and TDD (average 3.4 years); whereas, the novices were Master's students participating in an Extreme Programming course. The results show that experts are able to achieve better productivity (time to complete the task) but not quality (passing acceptance tests) for which a non-significant difference was found. Nevertheless, the authors conjecture that the observed difference might be due to the novice subjects' general lack of programming experience. Process conformance was measured, but as a separate factor from the developers' experience. The authors report that the experienced subjects adhered more to the process than novices, by a significant amount.