Information and Software Technology 000 (2017) 1–14

Contents lists available at ScienceDirect

Information and Software Technology

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

Findings from a multi-method study on test-driven development

Simone Romano[a,*], Davide Fucci[b], Giuseppe Scanniello[a], Burak Turhan[b], Natalia Juristo[c]

[a] University of Basilicata, Viale Dell'Ateneo 10, Macchia Romana, Potenza, Italy
[b] M3S, University of Oulu, Pentti Kaiteran katu 1, Oulu, Finland
[c] Facultad de Informatica, Universidad Politecnica de Madrid, Campus de Montegancedo, 28660 Boadilla del Monte, Madrid, Spain

ARTICLE INFO

Article history:
Received 26 September 2016
Revised 13 March 2017
Accepted 20 March 2017
Available online xxx

Keywords:
Ethnographically-informed study
Qualitative study
Test driven development

ABSTRACT

Context: Test-driven development (TDD) is an iterative software development practice where unit tests are defined before production code. A number of quantitative empirical investigations have been conducted about this practice. The results are contrasting and inconclusive. In addition, previous studies fail to analyze the values, beliefs, and assumptions that inform and shape TDD.
Objective: We present a study designed, and conducted to understand the values, beliefs, and assumptions about TDD. Participants were novice and professional software developers.
Method: We conducted an ethnographically-informed study with 14 novice software developers, i.e., graduate students in Computer Science at the University of Basilicata, and six professional software developers (with one to 10 years work experience). The participants worked on the implementation of a new feature for an existing software written in Java. We immersed ourselves in the context of our study. We collected qualitative information by means of audio recordings, contemporaneous field notes, and other kinds of artifacts. We collected quantitative data from the integrated development environment to support or refute the ethnography results.
Results: The main insights of our study can be summarized as follows: *(i)* refactoring (one of the phases of TDD) is not performed as often as the process requires and it is considered less important than other phases, *(ii)* the most important phase is implementation, *(iii)* unit tests are almost never up-to-date, and *(iv)* participants first build in their mind a sort of model of the source code to be implemented and only then write test cases. The analysis of the quantitative data supported the following qualitative findings: *(i), (iii)*, and *(iv)*.
Conclusions: Developers write quick-and-dirty production code to pass the tests, do not update their tests often, and ignore refactoring.

1. Introduction

Test-driven development (TDD) is an iterative software development practice within agile methodologies [1]. It requires developers to follow three phases: The *red phase* causes a shift in mindset from the test-last approach to the test-first approach, while the *green phase* only develops enough code to pass the tests and the *refactoring phase* focuses on design quality through refactoring operations and uses a set of regression test cases as a safety net. It is claimed that TDD leads to better code quality due to its focus on testing, and improves developers' confidence in their source code [2]. Based on this claim, some software organizations have been quick to adopt TDD, while others are still evaluating its benefits in terms of cost, quality, and productivity [3,4].

To assess TDD, a number of primary (e.g., controlled- and quasi-experiments) and secondary (e.g., systematic literature reviews) empirical studies have been conducted. Primary studies (e.g., [5,6]) have been quantitative in nature and have produced contrasting or inconclusive results [7]. The secondary studies summarize the empirical research results regarding TDD by aggregating, to a varying extent, the evidence from controlled experiments, quasi-experiments, and case studies [3,4,7,8].

TDD has been marginally investigated from a qualitative point of view and from the perspective of the developer [9,10]. Qualitative studies, unlike quantitative ones, inquire into the underlying reasons and motivations behind a given phenomenon [11]. Among the kinds of qualitative methodological approaches, an

* Corresponding Author.
E-mail addresses: simone.romano@unibas.it (S. Romano), davide.fucci@oulu.fi (D. Fucci), giuseppe.scanniello@unibas.it (G. Scanniello), Burak.Turhan@oulu.fi (B. Turhan), natalia@fi.upm.es (N. Juristo).

http://dx.doi.org/10.1016/j.infsof.2017.03.010
0950-5849/

ethnographically-informed study forces researchers to attend to the taken-for-granted, accepted, and un-remarked aspects of a practice, considering all activities as "strange" so as to prevent the researchers' own backgrounds from affecting their observations [12]. In this type of study, researchers immerse themselves in the study context, participate in the study (e.g., by joining in conversations, attending meetings, reading documents), and observe participants without prejudice or prior assumptions [13].

In this paper, we present the results of an empirical study involving students and professional software developers. We involved 14 graduate students in Computer Science at the University of Basilicata, and six professional developers with one to 10 years' work experience. We asked participants to work in pairs, each pair is composed of a driver and a pointer developer. The goal of our study is to gain insights into how developers apply TDD and deal with each of its phases. In particular, we sought to explore the values, beliefs, and assumptions that inform and shape the application of TDD and its phases. Given this motivation, our methodological approach can be characterized as ethnographic [14–16]. We asked the participants to perform an implementation task. In particular, they had to add new functionality to an existing software implemented in Java using Eclipse. This software is a complex, industrial-like case of which participants had some knowledge. The first author immersed himself in the study environment, participated in conversations, and asked the participants how they were applying TDD to perform the assigned implementation task. We collected information by means of contemporaneous field notes, audio recordings of discussions, and copies of the artifacts produced by the participants during the implementation. Fine-grained data about the participants' application of TDD was also gathered through an automated tool installed in the participants' integrated development environment (IDE). This tool is intended as an Eclipse plug-in and runs in the background without interfering with the IDE use. We analyzed TDD conformance data to support or reject the qualitative findings. This is why we consider this as a multi-method study.

This study builds on a previous study [17] in the following ways:

- We described our ethnographically-informed study with more details.
- We extended the related work section by including a review of quantitative studies relevant for the approach we used to triangulate the results.
- We triangulated the initial qualitative results with analyses of quantitative data. We cross-referenced the results of both analyses and improved the discussion of the attained outcomes and conclusions.

In summary, we make the following contributions:

- We present the implications of the results from an ethnographically-informed study with nine pairs of developers; three of these pairs were professionals. To the best of our knowledge this is the first of such studies explicitly tackling TDD.
- We provide a triangulation of the results based on quantitative data extracted from the developers' IDE.
- We delineate future research to investigate the specificities behind the application of TDD.

The remainder of the paper is organized as follows. In Section 2, we discuss related work. In Section 3, we explain our method, while in Section 4, we present our findings. We show and discuss our findings in Section 5. In Section 6, we highlight limitations of these findings. Final remarks and future work conclude the paper.

2. Related work

In this section, we first discuss ethnographically-informed studies in software engineering and papers reporting quantitative and qualitative investigations on TDD.

2.1. Ethnographically-informed studies in software engineering

Ethnography is a qualitative research method for studying people and cultures. This method is largely adopted in disciplines outside software engineering [14]. However, the importance of ethnography, and the challenges associated with adoption from social sciences have been tackled in other areas of computer science, like computer-supported cooperative work [18]. In other subfields—e.g., software system design and interaction—ethnography is becoming progressively relevant [19]. In the context of software engineering, ethnography could provide an in-depth understanding of the socio-technological realities surrounding everyday software development practices [16]. That is, ethnography could help to uncover not only what practitioners do, but also why they do it. Despite its potential, little ethnographic research exists in the field of software engineering [16]. For example, Beynon-Davies [20] identified a number of uses of uses for ethnographic research in information systems development. In particular, the author noted that for researchers in the software engineering field, ethnographic research may provide value in the area of software development, specifically in the process of capturing tacit knowledge during the software life cycle. Later, Beynon-Davies *et al.* [21] used ethnographic research on rapid application development to uncover the negotiated order of work in a project and the role of collective memory.

Button and Sharrock [22] carried out an ethnographically-informed study on global software development to explain the knowledge that is displayed in the collaborative actions and interactions of design and development. Sharp and Robinson [23] reported on a study of eXtreme Programming carried out in a small company developing web-based intelligent advertisements. The main result being that the XP developers were clearly "agile." This agility seemed intimately related to the relaxed, competent atmosphere that pervaded the developers working in groups.

Singer *et al.* [24] found that software engineers maintain a large telecommunications system through habits and tool usage during software development. This study was hosted in a single company. Despite the software engineers stating that "reading documentation" was what they did, the study found that searching and looking at source code was much more common than looking at documentation. This case shows there is a difference between what practitioners say they do and what they actually do. Ethnographic research might help in highlighting and explaining such discrepancies to make clearer un-remarked aspects of practice [12].

Further, Salviulo and Scanniello [25] conducted an ethnographically informed study with students and professionals to understand the role of comments and identifiers in source code comprehensibility and maintainability. Authors observed the following outcomes: *(i)* professional developers (as compared with students) prefer to deal with source code and identifiers rather than comments, *(ii)* all participants (professionals and students) believed that the use of naming convention techniques when writing identifiers was essential, and *(iii)* all participants stated that the names of identifiers are important and developers should properly choose them.

2.2. Studies on TDD

Although ethnography has been used to study how testing is done within software companies (e.g., [26,27]), there is a lack of

Please cite this article as: S. Romano et al., Findings from a multi-method study on test-driven development, Information and Software Technology (2017), http://dx.doi.org/10.1016/j.infsof.2017.03.010