Test First, Code Later: Educating for Test Driven Development

Teaching Case

Naomi Unkelos-Shpigel(✉) and Irit Hadar

Information Systems Department, University of Haifa,
Carmel Mountain, 31905 Haifa, Israel
{naomiu,hadari}@is.haifa.ac.il

Abstract. As software engineering (SE) and information systems (IS) projects become more and more of collaborative nature in practice, project-based courses become an integral part of IS and SE curricula. One major challenge in this type of courses is students' tendency to write test cases for their projects at a very late stage, often neglecting code coverage. This paper presents a teaching case of a Test-Driven Development (TDD) workshop that was conducted during a SE course intended for senior undergraduate IS students. The students were asked to write test cases according to TDD principles, and then develop code meeting test cases received from their peers. Students' perceptions towards TDD were found to be quite positive. This experience indicates that instructing SE courses according to TDD principles, where test cases are written at the beginning of the project, may have positive effect on students' code development skills and performance in general, and on their understanding of TDD in particular. These findings are informative for both education researchers and instructors who are interested in embedding TDD in IS or SE education.

Keywords: Software engineering · Requirements engineering
Test Driven Development · Education

1 Introduction

In the last two decades, as the agile manifesto [2] has been increasingly adopted in industry, the contribution of collaborative projects has been recognized, focusing on continuous review among practitioners within and among development teams. However, information system (IS) and software engineering (SE) students are typically not being trained during their studies for this type of collaborative work [15].

Project-based courses, in which students are required to develop a prototype as their final assignment, are an integral part of IS and SE degrees' curricula. The teachers of these courses face several challenges, such as ensuring equal participation of all students in the workload [12], and creating projects that will be both of high quality and utility, all in parallel to teaching a large portion of theoretical background [15].

In recent years, collaborative practices have been used in SE education, in order to enhance student participation in tasks throughout the semester. This interactive method

R. Matulevičius and R. Dijkman (Eds.): CAiSE 2018 Workshops, LNBIP 316, pp. 186–192, 2018.
https://doi.org/10.1007/978-3-319-92898-2_16

of experiencing other groups' work, while presenting their own, resulted in positive feedbacks about the projects and the assessment method [15].

Test-driven development (TDD) is a software development practice, encouraging developers to write tests prior to code development [3]. This practice has become popular in recent years in industry as a requirements specification method. However, several challenges still hinder TDD practices in both industry and education [4].

This paper presents a teaching case of a SE project-based course for IS students. Leveraging on existing examples for TDD in practice, this paper presents a teaching case of incorporating a TDD workshop into this course. While designing and executing this workshop, the following research questions arose: (RQ1) How can we provide the students with an experience that will emphasize the advantage of using TDD over traditional testing? (RQ2) How do students perceive TDD following this workshop?

The next section presents the background for our research. Section 3 details the TDD workshop. Section 4 presents the findings, and Sect. 5 discusses the conclusions and future research directions.

2 Background

2.1 Test Driven Development (TDD)

Test driven development (TDD) is a software development practice, encouraging developers to write tests prior to code development [1, 3]. In recent years, TDD has become very useful in specifying the desired software behavior [7]. As explained by Hendrickson [7]: "The TDD tests force us to come to a concrete agreement about the exact behavior the software should exhibit."

Developers who practice TDD contribute to product quality and overall productivity [4, 5]. In addition, when examining the use of TDD in academic environments and in the industry, a major improvement in product quality is observed [3, 9].

A major challenge of TDD is that it does not provide testers and developers with full guidance on which parts of the code they should focus. Instead, they are expected to understand on their own, which parts of the code they should test, and investigate why these tests fail [11]. Another challenge is related to the collaborative nature of the TDD practice; while TDD has been found to increase collaboration among developers and testers in lab environments [4], further research is needed in order to understand how collaboration contribute to practicing TDD in industry-scale projects.

2.2 Collaborative Requirements Engineering (RE)

A major concern of the current RE practice is collaboration [5, 8]. Using agile practices has led to solving many challenges of traditional RE. However, performing RE according to the agile principles is still an evolving craft. Inayat et al. [10], conducted a systematic literature review in order to understand what the main characteristics and challenges in agile RE are. Out of over 540 papers retrieved, only 21 were found to be relevant to agile practices. A key issue in the relevant papers is requirement change management, as requirements elicitation in agile development is usually based on user

stories, which are frequently changing. Therefore, requirements are validated usually through prototyping, and documentation hardly takes place [10].

Several research works examined the practice of collaborative TDD. Romano et al. [11] used ethnographic research in order to study novice programmers' reaction to performing TDD in pair programming. They concluded that this practice holds several limitations to collaboration and requires tool support. They also made an interesting observation that programmers first visualize the code, and only then write the test. An additional research, which analyzed TDD practices in GitHub, found that TDD may increase collaboration among testers and programmers, but may also hinder the process, as it sets a high bar of expected results [4].

3 The TDD Workshop

The TDD workshop was designed as a teaching method, executed in two consecutive years in an advanced SE course, intended for third year undergraduate IS students. The course consisted of 59 students in fall 2016-7, and 75 students in fall 2017-8. The students worked in teams of four, building a software product. In the first three weeks of the semester, the students wrote a requirements document and constructed a rapid prototype. Following these tasks, they developed the product employing SCRUM [13] methodology.

In order to teach the students TDD, we conducted a short lecture explaining TDD and its practice. Next, we conducted a two-part workshop, as explained below.

In the first year, the workshop took place in the 6th and 7th lecture. In the first week of the workshop, the students received an assignment (see Fig. 1) and were asked to write as many automated tests as they could think of. The groups uploaded the tests into the course's website. In the second week of the workshop, each group received tests written by another group, and were asked to code functions accordingly.

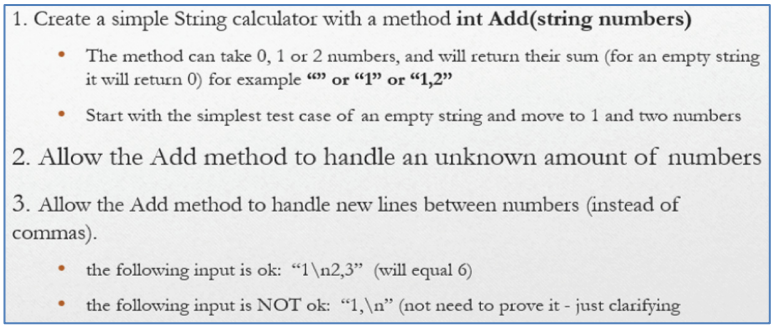

Fig. 1. The task given in the first workshop (taken from: http://osherove.com/tdd-kata-1)

In the second year, the procedure was repeated with the following two alterations: (1) the workshop took place in the 2nd and 3rd lecture. (2) The task was different - In the first week of the workshop: the students received the task of finding valid URLs in an

online chat history messages. The reason for the first change was that we wanted to see whether performing the workshop at an earlier stage in the course, would affect performance. The second change was aimed to enable the students to face a "real life" coding challenge, which they are familiar with from their online activity.

4 Students' Reflections on the TDD Workshop

At the end of the course, the students filled out a questionnaire, in which they were asked about their opinions regarding each part of the workshop. In addition, at the end of the semester, the students were asked to describe their perceptions on TDD and its challenges, based on their experience in the workshop. Figure 2 presents the main categories found in each semester (Fall 2016-7 – 52 respondents, Fall 2017-8 – 65 respondents). We analyzed the responses inductively [14] with regards to the research questions.

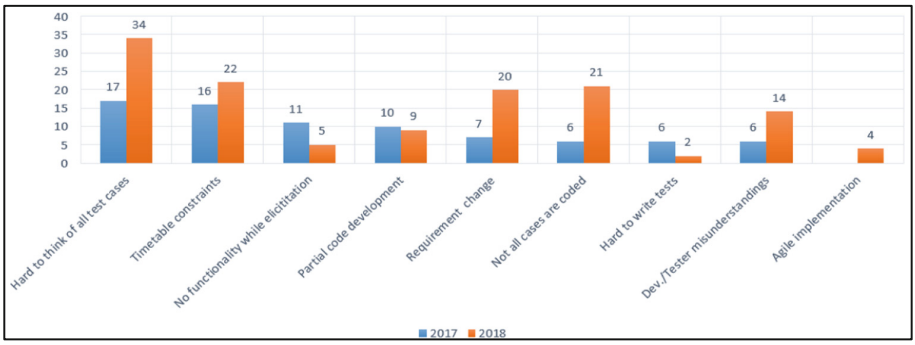

Fig. 2. TDD challenges emergent categories

RQ1 – The students referred to TDD as an interesting and beneficial, as well as a challenging experience. Referring to the first part (tests writing), they mostly addressed the challenge of writing tests without having a product or code:

- *"Very challenging – forced us to think of all possible tests"*
- *"It taught us how to deal with unfamiliar code, and how to test it"*

They also mentioned the relatedness of the task to the course project:

- *"Very good. We divided the work among team members, which was a good preparation for the project."*

Referring to the second part (code writing), the responses mostly addressed the challenge of writing code according to tests received from other groups:

- *"It was very hard to write code according to other students' tests"*
- *"Great experience! it gave us an idea of how the situation will be in the industry, because it's always going to be this way: we will get some code and we'll have to make some changes on it."*

- *"It was very surprising when we got to code test [cases], which we did not think of in the first part."*

When addressing both parts of the workshop, some students mentioned the importance of this practice to their future careers. Others addressed both parts of the task as informative and helpful in viewing tests and code tasks through the perspective of other groups.

RQ2 – In both semesters, the students addressed some challenges of TDD, which have already been discussed in the literature [6]. These include the difficulty to achieve requirements coverage, the required ongoing communication between testers and developers, and longer development duration. In the second semester, when the students performed the workshop as a requirements elicitation procedure for their project, they focused in their responses mostly on requirements coverage and requirement changes.

Some interesting observations emerged from analyzing the full responses:

- In both semesters, students addressed the TDD challenges of partial code development, longer development, the difficulty of writing tests prior to development, and the problem of eliciting tests with no product or code present. The challenge of longer development time was the one most frequently mentioned.
- Students in the second semester addressed the problem of tests' coverage ("it is hard to think of all tests") about twice as much as students in the first semester. This finding could be related to the task itself, which was more complex in the second semester, and required thinking of many possible faults (erroneous URL structure, website not found, etc.).
- The topics of requirements coverage and change were addressed much more frequently in the second semester (about three times as much). We believe that the fact that students in the second semester faced TDD while they were in the process of eliciting requirements for their project, made them more aware to the aspect of TDD as requirements specification method. Furthermore, as they faced requirements change throughout the duration of the semester, at the end of the process they were experienced at coping with these changes. They realized that tests written at the beginning of the process could not cover these changes.
- The challenge of misunderstandings between developers and testers was also mentioned more frequently (about twice as much) in the second semester. As the task was more complex in the second semester, this finding was to be expected.
- A new category, which emerged only in the second semester, is the difficulty of conducting TDD in agile projects. Since TDD is part of agile practices, this category is surprising. Students explained this challenge with the fact that agile development requires prompt results. Performing TDD "by the book", usually requires spending a substantial amount of time in the beginning of each development cycle. This is perceived as delaying code development.

5 Expected Contribution and Future Directions

In this paper, we presented a teaching case, designed to enhance students' perception and performance of TDD practices. According to our findings, performing the workshop at the beginning of the course, resulted in students enhanced understating of TDD's advantages and challenges. This finding can assist and guide instructors who are interested in embedding in their course TDD as a requirements specification method. We intend to repeat this workshop next year as well, and to add TDD tasks to the course project. This will enable to evaluate the overall project quality while performing TDD. Such an evaluation will add quantitative performance indicators to the students' self-reported perceptions elicited in this exploratory study.

References

1. Beck, K.: Test-Driven Development: by Example. Addison-Wesley Professional, Boston (2003)
2. Bissi, W., Neto, A.G.S.S., Emer, M.C.F.P.: The effects of test driven development on internal quality, external quality and productivity: a systematic review. Inf. Softw. Technol. **74**, 45–54 (2016)
3. Bjarnason, E., Sharp, H.: The role of distances in requirements communication: a case study. Req. Eng. **22**(1), 1–26 (2015)
4. Borle, N.C., Feghhi, M., Stroulia, E., Greiner, R., Hindle, A.: Analyzing the effects of test driven development in GitHub. In: Empirical Software Engineering, pp. 1–28 (2017)
5. Fucci, D., Turhan, B., Juristo, N., Dieste, O., Tosun-Misirli, A., Oivo, M.: Towards an operationalization of test-driven development skills: an industrial empirical study. Inf. Softw. Technol. **68**, 82–97 (2015)
6. Gopinath, R., Jensen, C., Groce, A.: Code coverage for suite evaluation by developers. In: Proceedings of the 36th International Conference on SE, pp. 72–82. ACM (2014)
7. Hendrickson, E.: Driving development with tests: ATDD and TDD. Starwest (2008)
8. Marczak, S., Damian, D.: How interaction between roles shapes the communication structure in requirements-driven collaboration. In: 2011 19th IEEE International Requirements Engineering Conference (RE), pp. 47–56 (2011)
9. Kurkovsky, S.: A LEGO-based approach to introducing test-driven development. In: Proceedings of the 2016 ACM Conference on Innovation and Technology in Computer Science Education, pp. 246–247. ACM (2016)
10. Inayat, I., Salim, S.S., Marczak, S., Daneva, M., Shamshirband, S.: A systematic literature review on agile requirements engineering practices and challenges. Comput. Hum. Behav. **51**, 915–929 (2015)
11. Romano, S., Fucci, D., Scanniello, G., Turhan, B., Juristo, N.: Results from an ethnographically-informed study in the context of test driven development. In: Proceedings of the 20th International Conference on Evaluation and Assessment in SE, p. 10. ACM (2016)
12. Schneider, K., Liskin, O., Paulsen, H., Kauffeld, S.: Media, mood, and meetings: related to project success? ACM Trans. Comput. Educ. (TOCE) **15**(4), 21 (2015)
13. Schwaber, K.: Agile project management with Scrum. Microsoft Press, Redmond (2004)

14. Strauss, A., Corbin, J.: Basics of Qualitative Research: Techniques and Procedures for Developing Grounded Theory. Sage Publications, Thousand Oaks (1998)
15. Unkelos-Shpigel, N.: Peel the onion: use of collaborative and gamified tools to enhance software engineering education. In: Krogstie, J., Mouratidis, H., Su, J. (eds.) CAiSE 2016. LNBIP, vol. 249, pp. 122–128. Springer, Cham (2016). https://doi.org/10.1007/978-3-319-39564-7_13