Appreciate the Journey not the Destination - Using Video Assignments in Software Testing Education

Adnan Čaušević
Software Testing Laboratory, Mälardalen University
Västerås, Sweden
adnan.causevic@mdh.se

***Abstract*—Courses related to software testing education, at the university level, in most cases have a learning outcome requiring from students to understand and apply a set of test design techniques upon completing the course. The problem, however, remains on how to both effectively and efficiently evaluate if a student has accomplished the stated outcome. By purely looking at the final resulting set of the test cases provided by a student, it is not evident which, if any, test design technique was used to derive them. In this paper, we are presenting a rather simple but effective method of collecting video assignment submissions from students instead of a traditional source code and tests solution. This way, the teacher could rather quickly and in detail gather evidence that student indeed obtained the knowledge needed for passing the stated learning outcome.**

I. Introduction

With an ever-growing number of university students, who are studying software engineering, the need for efficient and effective methods of evaluating homework assignments is evident. By efficient, here we are mostly referring to the effort spent on grading these assignments by the examiner (or the teacher) and such an effort is often expressed in the time needed to complete it. In addition, another type of resources could be spent in grading, like using specific hardware where assignment has to be downloaded or a specific service (as in cloud-based) that has to be accessed. When referring to the effective evaluation of homework assignments, here we are mostly interested in how to easily detect deviations between the provided assignment and a given learning outcome for which the assignment was designed.

Software Testing education, as an important aspect of software engineering, can not be isolated from this problem. Practical assignments, in which it is often asked from students to apply specific test design techniques on a given system under test, could rather easily grow in complexity. But more importantly, in contrast to other disciplines, having only one correct solution in this field is rather unusual. This problem highlights the importance of evaluating how students applied a given test design technique rather than its outcome.

In this paper, we are presenting a rather simple but effective method of collecting video assignment submissions from students instead of a traditional source code and tests solution. This way, the teacher could rather quickly (by going fast-forward) and in detail (by pausing and replaying certain parts) gather evidence that student indeed obtained the knowledge needed for passing the stated learning outcome. In addition, by sharing videos among students via course social media channels, motivation for students to provide quality solutions is even higher.

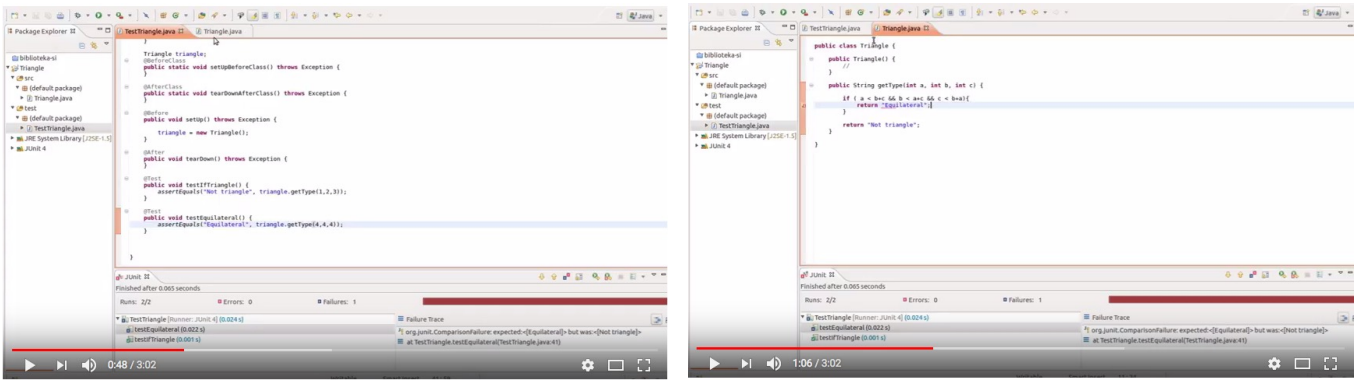
In the following section, we are discussing existing reports on advancing test education with respect to efficient assignment grading process, as well as some approaches to using video as a media for assignments. Section III address the peculiarities of software testing assignments and in particular discuss how to evaluate correct usage of the test-driven development approach. Section IV presents the process used for video assignments in one instance of a Software Verification & Validation course. Section V discusses threats to the general applicability of this approach with final conclusions and future work ideas presented in Section VI.

II. Related Work

When it comes to increasing efficiency of software programming assignments evaluation, several approaches have been proposed [1]–[5]. Most of these approaches are focusing on the automated grading of students code by utilising a back-end testing system. In some cases, evaluation is done on the tests as well in the form of coverage reports and also using static analysis as a mean to provide code styling conformance checks. But, most of these approaches are focusing on grading the final assignment submission of a student. As we discuss further in this paper it is evident that omitting the process, used by a student in creating the assignment, may hinder correct evaluation of obtained knowledge in applying test design techniques. Consequently, we are proposing the usage of video assignment submissions as a mean to overcome this aspect of the students grading.

Using videos as a media for student assignments has been already proven successful [6], [7]. Leah, for example, presented how students of Adelphi University have used video recording devices and video editing software to perform assignments related to their motor (physical) skills [6]. Kearney and Schuck [7] discuss in details the nature of student video projects and its relation to learning. Both students and teachers in their studies have indicated that video projects of students do develop authentic learning, referring to students ability to connect their education to the real world.

However, to the best of our knowledge, we have not seen the usage of video techniques in the grading of software testing education assignments. Also, it is important to note here



(a) A new test is added

(b) A new code is added

Fig. 1: Student writing a code to pass the previously failing test.

that we are discussing students video production using video recording devices (camcorders). In our case, when discussing video assignments, we are referring to computer desktop video recording production.

III. Software Testing Assignments

Learning outcomes presents an important instrument in ensuring the quality of a course at the higher level educational institution. For that reason, it is rather significant to properly design evaluation assignments and match them to the learning outcome.

For example, within the course on Software Verification & Validation, given at Mälardalen University, we have defined, among others, the following learning outcome:

> *Upon fulfilling the course, the student should be able to apply various software testing techniques in development projects.*

The idea is very simple, we expect from our students to know how to apply test design techniques. However, the problem arises when we try to evaluate if students have correctly applied a particular test design technique. Simple because our assignments are designed as follows:

> *Create a working Eclipse project for the Triangle problem.You need to provide both the program and the test cases.You have to create tests before the code (Test-driven Development approach).*

By providing us with the Triangle.java and TriangleTest.java files, we, the teachers, have no ability to validate if students indeed applied test-driven development approach and what guidance they used when designing the tests. There is an evident need to have a more detailed view of the actual process conducted by students. In the following section, we discuss further why is this concern important in a test-driven development education.

A. Grading the TDD process

Test-driven development process introduces an interesting paradigm for software development [8]. By enforcing developers to write tests prior to the code, in small time cycles, developers are constantly building quality into their code. However, the process itself is rather sensitive to deviations and thus should be followed in a rather strict "red-green-refactor" patter.

- Red: The developer should first write a test and execute it to see it fails, and thus visualise that there is a problem which needs to be fixed.
- Green: Afterwards, the developer has to create a minimal code needed to make the previously failing test pass. Only once all the tests are passing, the developer can proceed to the following step.
- Refactor: After few iterations on the code it is important to refactor and improve the design of code. No new functionalities are added and existing test cases are ensuring that the current functionality is not corrupted.

Although not a direct test design technique, TDD does represent an important aspect of overall code quality. It is necessary to have the students exposed to it in order to better appreciate the effort of proper test design as well as test automation. However, due to the fact that the TDD concept is new to students and sometimes even considered rather unorthodox, we need to make sure that the students are indeed following it properly.

IV. Video Submissions

In this section, we are presenting our experience from applying the video assignment approach to an instance of the Software Verification & Validation course, given at Mälardalen University in 2017. At the very beginning of the course, once the students are introduced to the fundamentals of software testing and in particular the test case design when using the test-driven development approach, an assignment was given to them in the form of a video submission. They were expected to record their desktop screen with Eclipse[1] integrated development environment being open for the whole duration until the assignment is completed. The video is then to be posted on an internal social network at Mälardalen University

[1] www.eclipse.org

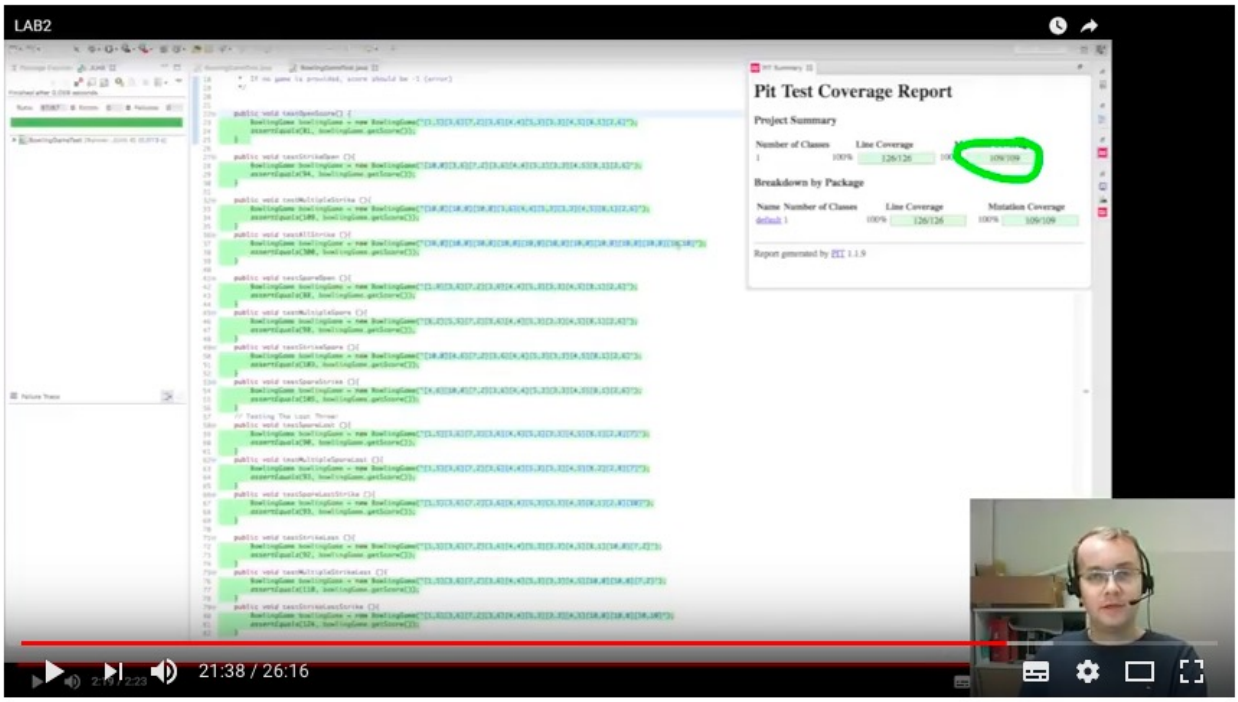


Fig. 2: Teacher using *telestrator* while providing video feedback to students

(Yammer[2]) within a group dedicated to students of this course. Figure 1 depicts a screenshot of one student submission as an example. Although students did not expect this type of assignment submission and had not previously done anything similar to it, they still managed to get the videos posted on time and with a rather good quality. Many of the solutions were also correct, but for a few, the teacher had to further discuss students' submissions with them.

At least one of the students made a mistake by writing first code and then tests. When asked about this, the student admitted that he completely forgot that the assignment should be done by following the TDD approach, and rather soon re-submitted his new video solution, which was correct this time. However, the most interesting issue that was discussed with students, was the interpretation of the following TDD rule:

> *the developer has to create a minimal code needed to make the previously failing test pass.*

On several occasions, it was noticeable that students are not really following this rule and as soon as they have a failing test, they write a code that goes way beyond the scope of that test. This results in having written a code that was not fully tested. This was discussed in details with students in one of the follow-up lectures after the assignment deadline. Essentially, their inexperience with TDD has led to making such a mistake. Still, having video assignments, it made it easier for teachers to detect this deviation and address it as early as possible such that incorrect application of TDD practice in future is hopefully prevented.

somebody cooked here

[2]www.yammer.com

A. Tooling

An important aspect of the success of video assignments is the tooling support for it. There exist several commercials, free or open source solutions for screen recording software. We did not impose on students the usage of any specific tool but rather suggested an existing open source solution that works on most platforms students could use without bringing any additional cost to them. The tool in question was Open Source Broadcaster (OBS).[3] Although several students did follow the suggestion to use OBS software, others tried out different software they sought more suitable for the job.

B. Video Feedback

Providing timely feedback to students assignments is an important factor in the success of a teaching method. To improve efficiency of this process, teachers have applied several rather interesting ideas. Plimmer and Mason proposed the usage of a paperless environment for annotation of students assignments [9]. Morris and Chikwa [10] evaluated if using audio comments instead of a paper based student feedback would increase the overall students' grades. Their findings pointed out no significant difference although students were positive regarding audio feedback.

To support students in creating videos, the teacher in the course has also made a few. Figure 2 picture a screenshot of a pre-recorded video feedback provided to students. Essentially it is a video of a teacher watching and commenting on other students videos. Our idea was that we cannot expect students

[3]www.obsproject.com

someone REALLY cooked here!

to do something that we, the teachers, are not ready to use ourselves. Video feedback was accepted by students as a rather positive approach in giving a timely feedback.

V. Threats to Applicability

In this section, we present two potential threats to a general applicability of using video assignments in software testing education.

One of the general threats to introducing new types of assignments at a higher education institution is the phenomenon of plagiarism. At Mälardalen University there is a recommendation to design assignments such that they discourage any possibility of plagiarism appearing. However, as stated by Leah [6], students who submitted video assignments had to actually do them by themselves as they did not had a possibility to reuse already completed assignment from a previous year students. We noticed the same pattern in our study as students could not just copy & paste the code and test cases from another student pier. Instead they had to write first test and then the minimum code making that test pass. It would require a significant skill to be able to decompose an existing code and test solution into a TDD pattern (reverse engineering the TDD process). However, this is only applicable to the TDD related assignments. For any other type of video assignments plagiarism opportunities do exist and they should be treated accordingly. One approach would be to record a video camera stream overlay of the student during the completion of their task. However, this approach would increase the privacy issue and it remains unknown what effect it might have on the quality of the assignment.

Another interesting aspect to discuss regarding general applicability of video assignments is its usage by blind or visually impaired users. In previous instances of Software Verification & Validation courses we have witnessed challenges with respect to tooling support for blind and visual impaired persons. For example, one of the assignments includes code and test analysis using structural (code) coverage techniques. However, most tools for coverage provides colour-coded information regarding the state of a singe statement and/or branch. This rendered to be completely useless for some blind students that we had in a course. Indeed, this aspect has to be further investigated and better understand to what degree it is possible for blind and visual impaired users to perform desktop video recording.

VI. Conclusion and Future Work

Homework assignments are classified as *active learning* type of approach to knowledge building process. This is where students have to *actively* work on solving a dedicated practical problem given to them. However, they have to do that alone with no immediate help from the teacher or other peers in the course. In order to truly evaluate if the homework assignment has been conducted correctly, the teacher needs to *see* the process of creating it, rather than focusing on the actual solution or its outcome.

In this paper, we have presented an approach of using video assignments in grading students submissions, especially within the software testing education. By having a video submission from students, the teachers can potentially better utilise their time when evaluating if a solution for a given assignment fulfils the learning outcome of the course. Within the course of Software Verification and Validation at Mälardalen University, this approach was tested by applying it to the *TDD assignment*. As a result, it was noticed that students to some extent do not follow the rules of TDD and could potentially accommodate the wrong process for using it. With the help of video assignments, this was noticed early in the course and students had the time to discuss and reflect on the mistakes done.

For the next instance of the course, the plan is to integrate video assignments within the learning platform itself rather than using social platform for it. In addition, a short introductory video could be made for students on how to create their own videos and motivate why this way of working is beneficial for students as well as teachers.

Acknowledgement

This work was supported by the Swedish Knowledge Foundation (KKS) through projects FuturE and PROMPT - Professional Master's in Software Engineering. The author would like to thanks to the colleagues and students at Mälardalen University for their inspiration and support.

References

[1] S. Elbaum, S. Person, J. Dokulil, and M. Jorde, "Bug hunt: Making early software testing lessons engaging and affordable," in *Proceedings of the 29th International Conference on Software Engineering*, ser. ICSE '07. Washington, DC, USA: IEEE Computer Society, 2007, pp. 688–697.

[2] J. Spacco and W. Pugh, "Helping students appreciate test-driven development (tdd)," in *Companion to the 21st ACM SIGPLAN Symposium on Object-oriented Programming Systems, Languages, and Applications*, ser. OOPSLA '06. New York, NY, USA: ACM, 2006, pp. 907–913.

[3] S. H. Edwards and M. A. Perez-Quinones, "Web-cat: Automatically grading programming assignments," in *Proceedings of the 13th Annual Conference on Innovation and Technology in Computer Science Education*, ser. ITiCSE '08. New York, NY, USA: ACM, 2008, pp. 328–328.

[4] A. Allowatt and S. H. Edwards, "Ide support for test-driven development and automated grading in both java and c++," in *Proceedings of the 2005 OOPSLA Workshop on Eclipse Technology eXchange*, ser. eclipse '05. New York, NY, USA: ACM, 2005, pp. 100–104.

[5] I. Hernn-Losada, C. Pareja-Flores, and J. . Velzquez-Iturbide, "Testing-based automatic grading: A proposal from bloom's taxonomy," in *2008 Eighth IEEE International Conference on Advanced Learning Technologies*, July 2008, pp. 847–849.

[6] L. H. Fiorentino, "Digital video assignments: Focusing a new lens on teacher preparation programs," *Journal of Physical Education, Recreation & Dance*, vol. 75, no. 5, pp. 47–54, 2004.

[7] M. Kearney and S. Schuck, "Spotlight on authentic learning: Student developed digital video projects," *Australasian Journal of Educational Technology*, vol. 22, no. 2, pp. 189–208, 2006.

[8] K. Beck, *Test Driven Development. By Example (Addison-Wesley Signature)*. Addison-Wesley Longman, Amsterdam, 2002.

[9] B. Plimmer and P. Mason, "A pen-based paperless environment for annotating and marking student assignments," in *Proceedings of the 7th Australasian User Interface Conference - Volume 50*, ser. AUIC '06. Darlinghurst, Australia, Australia: Australian Computer Society, Inc., 2006, pp. 37–44.

[10] C. Morris and G. Chikwa, "Audio versus written feedback: Exploring learners? preference and the impact of feedback format on students? academic performance," *Active Learning in Higher Education*, vol. 17, no. 2, pp. 125–137, 2016.