Community Tests for Supporting Online Learning of Programming

Jozef Tvarozek, Peter Jurkovic

Abstract: *In the paper we focus on supporting online learning of programming through community-driven tests. We propose an approach to testing student solutions using tests created by students. The community tests simplify student work on solving exercises, and the learning system can automatically provide focused help to struggling students. We evaluated the proposed solution in an existing online learning system for programming. Results show that students used the new features often, and when examining the impact on learning performance, moderately active students had better performance than inactive student, while using the testing feature too much indicated worse performance.*

Key words: *Community test, Community Support of Learning, Online programming learning.*

1. INTRODUCTION

Education technology has been changing over the years in order to find most effective ways of learning. Universities also embrace new forms of education, and many have created their own online educational systems that support their large student groups in learning. Social presence and a sense of community have been found to be important for effective learning and knowledge building [3]. Community is a social unit that shares common values. A learning community [1], such as a school or class, is characterized by willingness of its members to share resources, accept and encourage new membership, regular communication, systematic problem solving and preparedness to share success. An online learning community [4] is a group of people, connected via technology-mediated communication, who actively engage one another in learner-centred activities to intentionally foster the creation of knowledge, while sharing a number of values and practices [6].

One way of including community in learning is through user-created content, which is being more and more included as a feature of online education because it encourages deeper engagement in learning. Teachers are starting to adopt supportive role while students are adopting roles of producers and commentators. The best known social software tool allowing knowledge creation using user-generated content is Wiki. Using a Wiki page, users can publish new content directly to the Web, including text, images and hyperlinks but also can edit already existing content. Despite some initial difficulties with adapting to this tool, Wiki pages were proven useful in creation of subject-specific knowledge bases [8]. There are however obstacles when involving community: motivation and passivity. When members of the community are not motivated enough active the benefits of being in it are lower. So not only current members are not contributing, but also the potential new members won't have a reason to join the community. Often, community members become members only to gain something. They want others to help them but offer no help in return. And if there are not enough people willing to help, the whole purpose of community is not fulfilled. It's also quite common for a community to form a core of some sorts, which consists of few individuals who are experts and are the main source of knowledge in the community. When this happens, there is a big risk that when someone from the core leaves or is not available, the communication in the community becomes much less frequent [2].

In our project, we are interested in engaging community in learning programing online. That is, we want to make use of effort each individual student is making, and use it to help others. In online learning of programming, learner's performance is typically rated by evaluating their programs, which are usually tested automatically. Before submission, students usually prepare test inputs and (corresponding) outputs for testing their own submissions. We aim to make use of the tests that individual students create, and have the tests reused (in appropriate situations) for other students.

We designed a solution that: 1) identifies the most appropriate test case for each struggling student, and that 2) makes it for students very easy to use the testing feature with only a single click. Given that hundreds of different tests cases are available, for students to use the testing feature effectively, it is necessary to select appropriate test cases individually.

2. INVOLVING COMMUNITY INTO LEARNING

In typical online learning of programming, students solve programming exercises and their answers are submitted for evaluation against test cases. In high stakes scenarios, such as course exams, the test cases may be kept hidden from students in order to prevent the students from hard-wiring correct outputs of the test data into their answers. Provided that the test data is open to students, they can examine their incorrect answers and try to repair them based on the test cases, and resubmit corrected solution. Whatever the programming exercise, however, teachers (content authors) cannot possibly cover all aspects of the exercise's problem description using tests, and therefore the test data is often times actually insufficient to completely evaluate correctness of students' answers. Nonetheless, even insufficient test data is usually useful for learning.

Many students prepare their own test cases for testing of their own answers before submitting. We have designed a solution for online learning of programming that facilitates: 1) test creation and test execution for each individual student and, 2) reuse of tests created by other students. Considering a medium-sized course with hundred students, each producing some two or three test cases of average, we easily can have 200 to 300 test cases available. It would not be very effective for every student to analyse answer (source code) against each of these test cases, but it is important to select appropriate test so that efficient learning can occur. Based on the zone of proximal development, we select the test case that should be the easiest for the particular student to repair.

The proposed solution is an extension of an online learning system for programming [7] which is similar to learning systems used at other institutions. In each course, students solve weekly programming exercises in an online code editor. Students can write code, submit the code for evaluation against hidden test data, and resubmit until the solution is accepted. In this proposal, we extend the system with a use testing facility that helps students write correct solutions – the code editor is extended with test status list (Figure 1), which contains the current status (OK, FAIL, or N/A) of each test that is available to the student against the current source code. Each student may have a different set of test cases available.

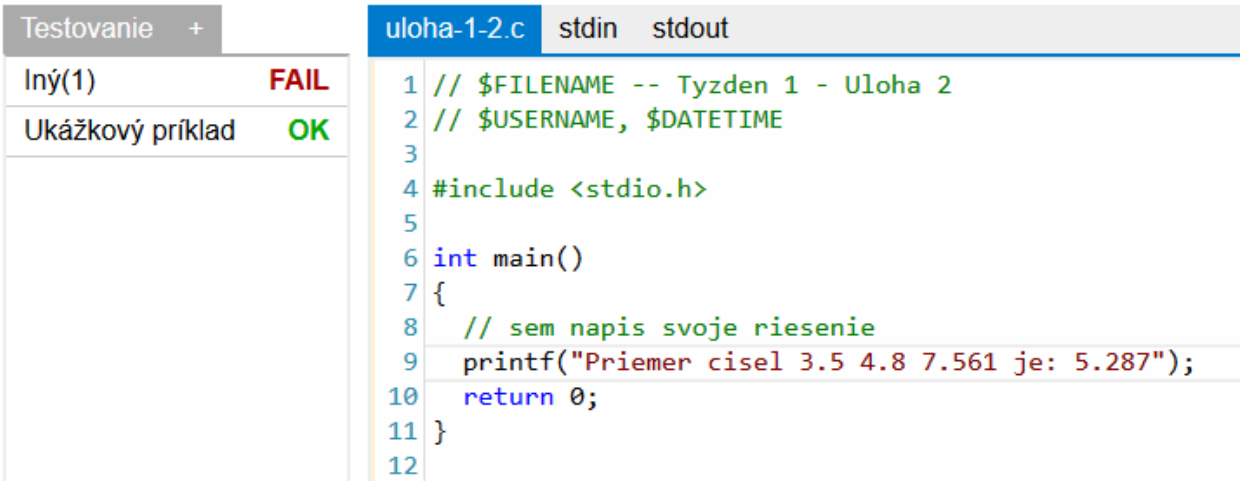

Figure 1. Current status of testing (left) and source code editor (right)

Student can create a test by clicking the plus sign and filling in the test's title, input data and expected output data for the test. Student activates a test with a simple click into

the test's row, and the current source code is immediately retested in a sandbox environment on the server, with the testing result updated in the web page. In case the test fails, that is the program's provided output is different to the test's expected output, a difference analysis (Figure 2) is provided to facilitate locating the error.

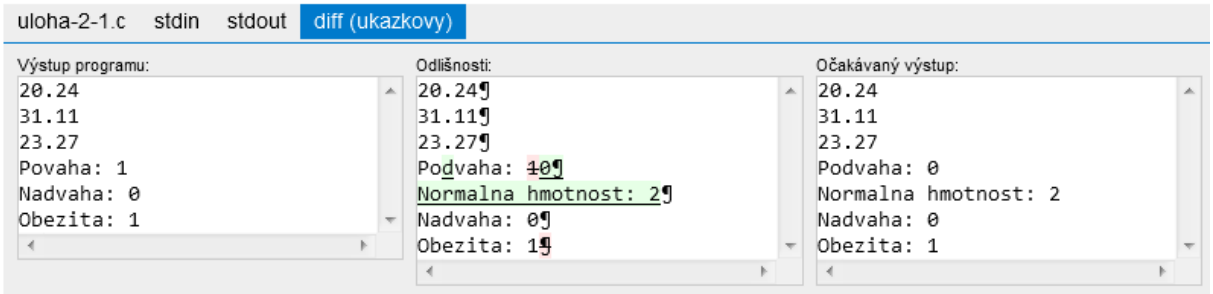

Figure 2. Analysing differences in provided output vs. expected output.

Tests that students create are reused for other students to help them find errors in their answers more quickly. When a struggling student submits a source code for evaluation, and the submission is evaluated as wrong (fails on some hidden test), our method selects a community test from all the tests available, such that the test selected is the most easiest for the struggling student to correct (repair in the student's current solution). Each student may, therefore, have different tests available for testing (Figure 1) depending on the pattern of incorrect answers he/she submitted.

For each available test, we keep statistics on how well the test is performing – passing/failing based on whether the corresponding source code (answer) was evaluated as correct or wrong – we keep four counters: fail_wrong, fail_correct, ok_wrong, ok_correct for each possible combination; e.g. fail_wrong is the number of submissions for which the test failed and the submission was evaluated as incorrect, fail_correct is the count of submissions for which the test failed while the submission was evaluated as correct (the test is either invalid, or is testing an aspect of the problem not covered in the hidden test data used in evaluation), etc.

The counter ok_wrong is the most interesting one, because it tracks the number of submissions which were evaluated as incorrect, but the test has passed. Provided that the test is according to the problem's specification, this statistic (ok_wrong) provides us with information on how difficult the test is to pass.

In our proposal, after a struggling student submits an incorrect answer, we select the test with the highest (ok_wrong/submissions_count) ratio that is not yet available for the student in the test status list. That is, we make available (publish) the easiest-to-pass test such that the student's submission is not yet passing the selected test. In this way, we aim to support the struggling student with a hint – community-based test – that is the most easiest to repair in student's current submission.

3. EVALUATION AND RESULTS

We evaluated the proposed method in an online learning system for programming [7] in a programming course on algorithms with 105 students actively participating. We evaluated how students use the testing facility – test creation, test usage, and test publishing – and how it affected student performance. The students used the proposed method for 5 weeks during the course, and were given 15 programming exercises to solve.

Test creation. Total of 386 tests were created (avg. 3.67 per student). 71.4% students created at least one test, 5.14 per student. An average of 25.73 tests created per exercise, and time-wise: average of 7.14 tests created per day. Test creation was relatively consistent each day, while most tests were created when there was a course's deadline on the exercises. There was not an even distribution among authors of tests: overall only 16 students created more than half of all tests.

Test usage. Total of 24 375 test evaluations were performed during the study duration (avg. 232.14 per student). 96.2% students used at least one test. On average, 451.39 tests were evaluated per day. We also analysed the type of the tests that students used to run – teacher-created example tests (22.5%), own tests (14.1%), published tests (63.4%). Figure 3 contains a breakdown per day. We can see that students used the tests that were selected by the method extensively.

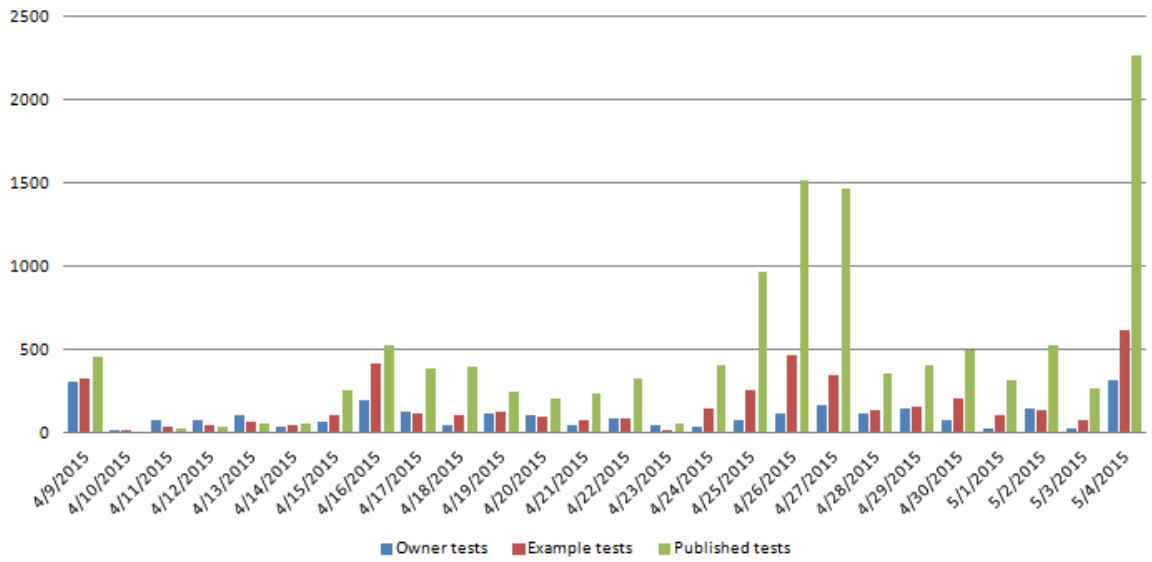

Figure 3. Breakdown of test usage per day and per type.

Test publishing. Total of 1636 wrong submissions we recorded during the time, and 54.9% of the time we could select an appropriate test for the struggling student. 130 out of the 386 tests that were created were also published at least once to another peer student, and 54.6% test creators (students) had their test published. 88.5% students received at least one test, on average 34.57 tests were published (based on incorrect submission). We see that students are able to create tests that are valid and usable by others.

Effect of test creation on student performance. We examined how students who created tests performed in solving the exercises. Students who created no tests (30 students) had a success rate 45.6%, while students who created at least one test had success rate 62.4%.

Effect of test usage on student performance. We examined how students usage of the testing facility resulted in better performance on solving the programming exercised. We computed average test usage per task, and based on that compared students divided roughly in half: the success rate in the group that used fewer tests was 52.9%, while the success rate in the group that used more tests was 62.8%. Within the groups, the success rate varied, for breakdown see Figure 4. Overall, the success rate slightly increased with increase in test usage.

Effect of received tests on student performance. Lastly we look on how the number of received tests affected student performance. Comparing those who did not participate and those who did shows significant difference: No-tests-received (group size 12) – 27.7 % success rate, Some-tests-received (group size 93) – 61.4 % success rate.

When examining more closely the students that received tests (Figure 5), we can see varied performance similar to the previous examination. In the class, we have observed that students were tricking the system to pass already available tests by hard-wiring the expected output so they can receive more. Similarly, the success rate slightly increased with increase in the number of tests student received.

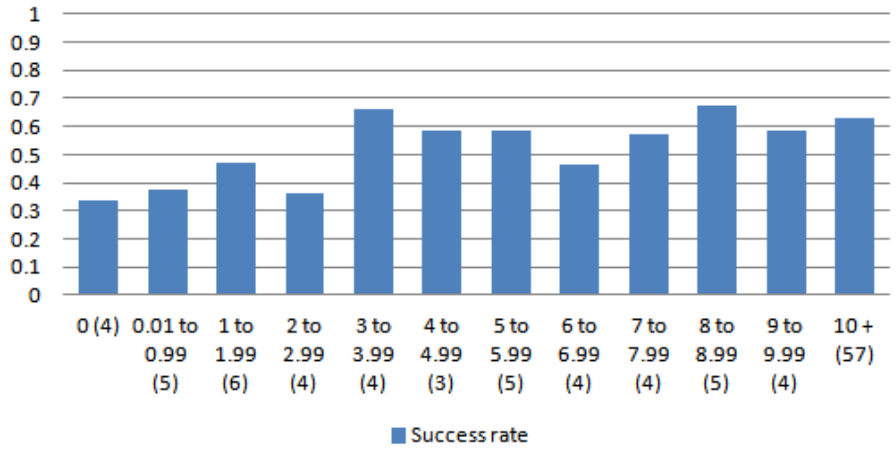

Figure 4. Effect of average number of test usages on success rate.

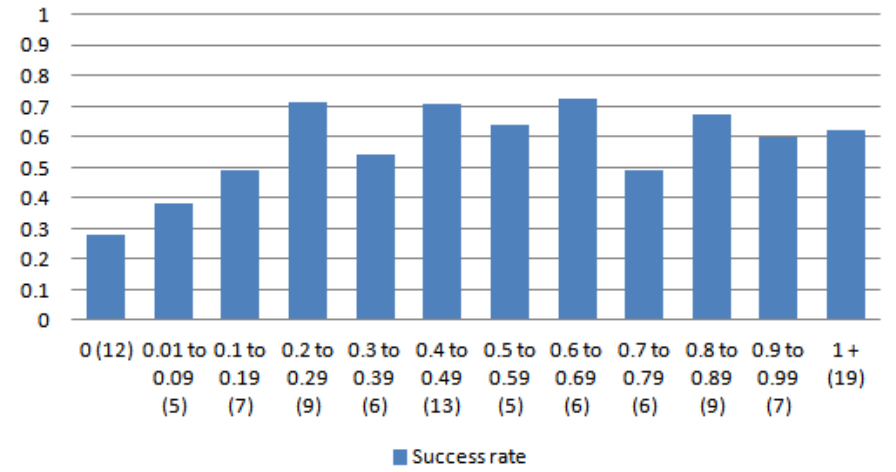

Figure 5. Effect of average number of received tests on success rate.

4. CONCLUSIONS AND FUTURE WORK

The goal of our project was to improve online programming learning through involvement of community of peers. We proposed a novel approach to testing – community tests – tests are created by students and they are published automatically to struggling students based on submission's error. The tests can be executed very easily with a single click, and differences between the expected and actual output are visualized to facilitate debugging.

In order to evaluate the proposed approach we collected data on 105 students using the community testing feature in an online programming course on algorithms. The results indicate that the new feature was indeed used by majority of the students and was also well received. Students were able to create valid tests with varying difficulty that enabled the system to provide other students with appropriate tests when the submission was wrong. Effects on learning performance appear to be positive: students who used community tests demonstrated better performance compared to those who did not use them at all. However, the most active students were less successful than the moderately active students.

In future, we plan to improve the method for determining the most useful test by allowing students to rate and comment the tests they received. Comments could also serve as useful user-generated hints.

ACKNOWLEDGEMENTS

This publication was partially supported by project KEGA No. 009STU-4/2014, and it is a result of Research & Development Operational Programme for the project ITMS 26240220039 co-funded by the ERDF.

REFERENCES

[1] Brook, C., Oliver, R. (2003). Online learning communities: Investigating a design framework. In: *Australian Journal of Educational Technology*, 2003, 19.2: p. 139-160.

[2] Fetter, S., Berlanga, A., Sloep, P. (2008). Strengthening the Community in Order to Enhance Learning. In: *Doctoral Consortium at the IADIS International Conference on Web Based Communities*, Netherlands: IADIS Press, 2008, p. 285-289.

[3] Gunawardena, C. (1995). Social presence theory and implications for interaction and collaborative learning in computer conferences. In: *International journal of educational telecommunications,* 1995, 1.2: p. 147-166.

[4] Kirschner, P. A., Erkens G. (2013). Towards a Framework for CSCL Research. In: *Educational Psychologist*, Routledge, 2013, 48.1: p. 1-8.

[5] Navrat, P., Tvarozek, J. (2014) Online programming exercises for summative assessment in university courses. In: *Proceedings of the 15th International Conference on Computer Systems and Technologies*. ACM, 2014. p. 341-348.

[6] Shea, P. (2006). A study of students' sense of learning community in online environments. In: *Journal of Asynchronous Learning Networks*, United States: Sloan Consortium, 2006, 10.1: p. 35-44.

[7] Tvarozek, J., Brza, T. (2014). Engaging Students in Online Courses through Interactive Badges. In: *International Conference on e-Learning 2014*. p. 89-95.

[8] Wheeler, S., Yeomans, P., Wheeler, D. (2008). The good, the bad and the wiki: Evaluating student-generated content for collaborative learning. In: *British journal of educational technology*, Blackwell Publishing Ltd, 2008, 39.6: p. 987-995.

ABOUT THE AUTHORS

Jozef Tvarozek, PhD, Institute of Software Engineering and Informatics, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovicova 2, 842 16 Bratislava, Slovakia, E-mail: jozef.tvarozek@stuba.sk

Peter Jurkovic, MEng, Institute of Software Engineering and Informatics, Faculty of Informatics and Information Technologies, Slovak University of Technology in Bratislava, Ilkovicova 2, 842 16 Bratislava, Slovakia

The paper has been reviewed.