Real-world Testing: Using FOSS for Software Development Courses

Evelyn Brannock
Georgia Gwinnett College
1000 University Center Lane
Lawrenceville, Georgia 30043
678-939-9007
ebrannoc@ggc.edu

Nannette Napier
Georgia Gwinnett College
1000 University Center Lane
Lawrenceville, Georgia 30043
678-524-1511
nnapier@ggc.edu

ABSTRACT
We designed a self-contained learning module on testing and free and open source software (FOSS) for a junior-level software engineering course. In this three-part module, students first learned software quality assurance concepts, and then used JUnit to create unit tests for their code. After being familiar with JUnit from a user perspective, students were required to investigate a defect reported in the JUnit code. Students were required to reproduce the problem, write test cases, and outline an approach for fixing the problem. In this pilot study, we conducted pre and post surveys of students' knowledge of and interest in FOSS. In the poster, visual data will be presented summarizing the results obtained.

Categories and Subject Descriptors
K.3.2 [**Computers and Information Science Education**]: Computer science education – *curriculum*

General Terms
Experimentation

Keywords:
Open Source, testing.

1. INTRODUCTION
Students are frequently motivated by real-world projects where their efforts can positively impact a client or broader user community [1-3]. For software development students, free and open source software (FOSS) projects provide an opportunity for authentic learning characterized by high relevance, ill-defined problems, and complex tasks [4]. Through active participation in an open source community, students can gain experience with large and complex codebases, learn a variety of software tools, and better understand the code behind the products they actually use. Seeking these benefits, some instructors have incorporated open source projects in capstone software engineering courses [5, 6].

SIGITE'12, October 11–13, 2012, Calgary, Alberta, Canada.
ACM 978-1-4503-1464-0/12/10.

Unfortunately, there are some challenges to getting started with FOSS development such as overcoming limited skills of faculty and students, identifying appropriate projects, and defining assignments that fit within the constraints of an academic semester. To address these challenges, we designed a self-contained learning module on testing and FOSS which required minimal pre-requisite knowledge of students. The module was used as part of a junior-level software engineering course.

2. CONTENT
Software Development I (SD I) requires a software quality module as a portion of its curriculum. Historically, the authors' software quality module consisted of two important components. The first (portion I) covered the important concepts such as non-execution based testing (walkthroughs and inspections), execution based testing (test plans and test cases), black-box testing and glass box testing, unit testing, integration testing, regression testing, writing test cases and plans and quality metrics. The second part (portion II) introduced the students to hands-on test-driven development that would use these concepts via the JUnit framework, a free and open source software (FOSS) automated testing tool that can be used with the Eclipse IDE. To learn how to use JUnit, the students were provided some pre-written Java classes, including a domain model class, a user view interface class, and a controller class and were asked to provide proper scripts to verify that all of the classes were individually valid, that they passed integration testing together, and all tests (according to the test data given) passed successfully. Any errors found were reported via a bug report and modifications required due to those errors were made. After comprehending the usefulness and efficacy of JUnit, a new exciting piece (portion III) was added to help the students to familiarize themselves with a "real-world" testing scenario by acting as contributors, in the course, to the JUnit FOSS community. They were now familiar with the operation of the JUnit software as a tester; they were challenged to contribute to the quality of JUnit's framework as a tester and bug fixer.

Their first directive as a tester and bug fixer for this actual, widely-used product was to read about the JUnit FOSS community to determine how an issue is reported, and to document the material an excellent, effective issue report should include. Then the students were assigned a REAL issue in JUnit (found at https://github.com/KentBeck/junit/issues). They were expected to determine if the issues was a bug or feature request, the date reported, and if it had been assigned. The next step was to reproduce the bug in their own environment. They were given a

little more additional information than the issue report gave, such as the fact that the instructor could reproduce the problem, and the exact version of JUnit to use to replicate the issue. The students were required to provide a minimum of two new additional test cases, one that provided correct results and one that provided incorrect results. The students were challenged to improve the original bug report, as the instructor purposefully chose an issue that omitted vital information (such as the operating system utilized and the exact version of JUnit.) A last requirement was to outline the approach for coding a successful solution for this issue including, after the fix is made, how their fix would be committed to the JUnit repository.

Fifteen students successfully completed the module, and were surveyed using pre and post tools. In the poster, visual data will be presented summarizing the results obtained. More than fifty percent of the students, in the pre and post assessments of the SD I course, recognized they had a high level of interest, and it was relevant to their career path. Interestingly, nine students in the pre-survey strongly agreed that "I like the mix of theoretical learning combined with hands-on application of the subject matter" as compared to only four in the post-survey. Also, data detailing the student's attitudes and acceptance of portion I of the module that addressed learning to use JUnit in an exercise and the portion II of the module that involved the students taking part in the development (in the quality assurance stage) of JUnit, as well as a comparison of the two portions, will be presented. The third prong of the module (portion III) in which the students were exposed to a

FOSS did appear to be more frustrating and difficult for the students. This was confirmed by anecdotal statements such as "for the FOSS testing project homework I would make sure that there are at least some concrete answers instead of having the questions require looking through hundreds of posts…", "a lot more thought goes into creating software than just coding", and "I really enjoyed the FOSS except that it was so unstructured I felt lost, which is very real world experience, but not at all satisfying…".

3. REFERENCES

[1] Pinkett, R.D., *Strategies for motivating minorities to engage computers*, in *Carenegie Mellon Symposium on Minorities and Computer Science*1999, MIT Media Laboratory.

[2] Schwartz, D., et al., *Towards the development of flexibly adaptive instructional design*, in *Instructional-design theories and models: a new paradigm of instructional theory*, C. Reigeluth, Editor 1999, Erlbaum: Mahwah, NJ. p. 183-213.

[3] Preston, J.A., *Utilizing authentic, real-world projects in information technology education.* SIGITE Newsl., 2005. **2**(1): p. 1-10.

[4] Reeves, T.C., J. Herrington, and R. Oliver. *Authentic activities and online learning*. in *Higher Education Research and Development Society of Australasia*. 2002. Perth, Australia.

[5] Morelli, R.A., et al., *Revitalizing Computing Education by Building Free and Open Source Software for Humanity.* Communications of the ACM, 2009. **52**(8): p. 67-75.

[6] Marmorstein, R., *Open source contribution as an effective software engineering class project*, in *Proceedings of the 16th annual joint conference on Innovation and technology in computer science education*2011, ACM: Darmstadt, Germany. p. 268-272.