Automatic Grading System for Programming Homework

Önder Demir, Aykut Soysal, Ahmet Arslan, Burcu Yürekli, Özgür Yılmazel
Computer Engineering Department
Anadolu University
Eskişehir, Turkey
{onderdemir, aykutsoysal, aarslan2, byurekli, oyilmazel}@anadolu.edu.tr

Abstract - One of the best methods of learning in programming courses depends on practical exercises. But preparing, collecting and grading homework manually takes time. When the number of students increases the number of homework given reduces. This reduces effectiveness of whole course. One way to solve this problem is distributing homework through an automatic grading system and getting back fast feedback.

This paper describes such a system that grades programming homework automatically. This system is using test driven software development methods and technologies that are being used by those methods. This system is tested with 8 homework in a course with 36 students and the results are discussed in this paper.

software testing; TDD; programming assignment; automated grading; unit testing

I. INTRODUCTION

Institutes that give computer engineering or computer science education, need to teach programming skills to their students. For this reason a suitable system for exercising programming must be implemented for students to try out their theoretical knowledge. One of the best ways to learning programming is to try to write programs that work. Homework provides a proper medium for students to write correct programs. Traditionally, homework are prepared and distributed via Internet, collected back, and then manually graded. This manual process both includes program output and source code analysis, thus takes time. This process works with small number of students, when the number of student increases, productivity decreases and human related errors increases on the instructor side. These problems reduce number of homework that can be given to students. Another problem with this process is that students are unable to see whether if they are on the right track or not to find a solution to the given homework.

The motivation behind this when designing such a system is to find a way both to ease the way students learn programming, and to provide a solution to prepare, distribute and collect homework quickly and efficiently.

When we express the problem, the first solution that came to our mind is that the solution could overlap with Test Driven Development (TDD) in many ways. TDD is a code development strategy that has been popularized by extreme programming [2, 4, 6]. In TDD the most important method is writing test before the code, and writing simplest code that can pass from the test, and repeating this cycle until the software is mature.

Using TDD in the classroom is not revolutionary [1]. Such automatic grading systems that uses TDD are used or still being used in many universities. The most important problem with systems that is based on TDD is that tests to grade homework need to written in a way that must lead and motivate students. For this reason, writing test can be cumbersome [1]. In order to reduce cumbersomeness we decide to use popular and well-documented open source TDD tools for our infrastructure. We develop a grading framework called Codepoint that uses jUnit [5] test framework as core. To distribute homework we use Maven [7], project management and build automation tool and for collecting homework, we use SVN [9] revision control software.

The rest of the paper gives background information in section 2 about challenges and TDD and in section 3, Codometer our automated grading tool is explained. The system is tested in a course having 36 students enrolled, with eight assignments. The results are discussed in section 4 with conclusion and future work in section 5.

II. BACKGROUND

A. Challenges

While we expressed the problem, we realized that there are some major problems which we have to solve:

- Students, especially freshmen, have difficulties to find the preliminaries of an assignment, struggle to create a workable solution of assignment.
- While solving the assignment students are not sure if they are in right track or not.
- On instructor side, human related errors increases while the number of given assignments increases.
- Both students and instructor have difficulties in distributing and collecting the given assignments.

The mentioned problems do not participate in learning cycle. They only reduce the productivity of a course and
learning process. While we express Codometer, we want to eliminate these problems.

B. What is Test driven development?

TDD is a methodology that is based on writing test before writing actual code [2]. While developing with TDD, only code that passes from tests is developed. After all the tests pass successfully, new tests are added to the system, and new code will be written to pass from those tests. This cycle is repeated and software developed evolutionary. While new tests are being developed, old tests are running to see that the code is in a stable state.

Writing tests first enables the developer to focus on the interface of the code without the distraction of the implementation details.

We can say that TDD has 3 main steps: [2]
1. Writing test.
2. Writing code that passes from the test.
3. Refactoring the code to remove duplicate codes and complexities.

C. Education and TDD

Automatic grading systems are used by many educators [3]. As TDD seems attractive, the idea of using TDD in the classroom is not revolutionary [1]. Computing and information technology educators have begun to call for the introduction of TDD into the curriculum [1, 8]. These systems generally depend on compiling and executing the code. One different system includes the test part in homework and checks how much those tests cover homework and grades according to that [3].

III. CODOMETER: A TOOL FOR AUTOMATIC GRADING SYSTEM FOR PROGRAMMING HOMEWORK

Codometer is a tool that distributes homework and grades the student code. Codometer has three main steps: educator distributes the homework with test packages, students commit their codes to the repository and Codometer automatically grades the committed codes. In classical announcement process of assignments, there are always problems such as not all students get or find the preliminaries of homework, students struggles to create working solution of homework which are not included in the learning process of computer science education. To overcome such problems we use Maven to distribute homework assignments. Before we start the Codometer project, in our requirement analysis phase we found out that, 37% of students are not sure if they understood the requirements of homework well. In another words they want to know if they prepared the solution of homework correctly. In order to solve this problem we package homework and use Maven’s dependency management infrastructure. Students can setup their working environment by adding the prepared homework package to their Maven project as a dependency. All homework packages include a command, so that students can execute the grading process and see their progress on the given homework.

A. Preparing Test Based Homework

To prepare a test based homework educator has to create a Maven project. In config file he has to declare dependency to Codepoint, grading tool of Codometer. Codepoint uses free testing tool called jUnit as core. With Codepoint educator can create countless test cases and assign points for assignments. Before every test method, educator has to put two annotations first @TestPoint(“<point for Test>”) and second @Test. An example of a test case is:

```java
@TestPoint(10)
@Test
public void TestCase()
{
    //body of test case must be written here
}
```

After preparing all test cases educator has to call “maven install” to create his test package. After successfully compiling, educator has a test package which can be used by students in their homework solutions. In this phase we suggest two steps of test cases. First step of cases will distribute to the students only if they are in right way to solve the homework. In second step of cases educator will use Codometer to do final grading. This second step can be a new set of test cases or also an extended version of first step cases or the same test cases of first step. After the deadline of homework instructor can easily change his first step with second step test cases or leaves in the same way and see the final grading of homework. By using first step test cases student will only have a brief idea if his doings are in right way. But also students are encouraged to do more tests on their own.
B. Distributing Homework

Educator has to put his test packages in a repository which is accessible by students. In the announcement, educator must announce the settings of the test package, the name and URL of repository, group id, artifact id and version of test package which are essential parts of a regular Maven project. An example of part of POM file is:

```xml
<repositories>
  <repository>
    <id>internal</id>
    <name>"Repository name"</name>
    <url>"Repository url"</url>
  </repository>
</repositories>
<dependencies>
  <dependency>
    <groupId>"groupID of test package"</groupId>
    <artifactId>"artifact id of test package"</artifactId>
    <version>"version of test package"</version>
  </dependency>
</dependencies>
```

When student creates a POM file that consists of settings which are announced all needed artifacts to create a successful solution will be gathered by Maven. By using Maven student does not struggle to create a correct solution, he can pay attention to only learning process of homework.

C. Instant Results of Homework

After student creates a successful solution, he must use the method startTests with his full qualified path of his package as parameter. While the test cases in package run, instantaneously student can see the output of tests and score which he gets. If he made a mistake during the implementation, he will simultaneously find the line number, the expected value, the error messages that Codepoint creates or the helpful hints that educator puts in test code to guide student in completing homework.

D. Collecting Homework

In classic case, each student has to deliver his homework solution to educator via e-mail, FTP or by physical media (such as CD, USB drive etc). This process works with small number of students, although when the number of student increases, productivity decreases and human-related errors increase. To address this problem we are using SVN, a source control tool. Each student commits his solution to SVN. SVN keeps track dates of commits. So even if a student commits homework after the deadline, with SVN educator can check out the homework before the deadline, thus providing a safe place for homework. After deadline Codometer collects all homework from SVN automatically. After gathering all homework the final grading step starts.

E. Final Grading

Last part of Codometer process is grading. After collecting all homework, Codometer starts grading process at a time which educator may set or by manual grade command by educator anytime. After grading finishes the results could be seen in Codometer user interface. Also results exported to XML files so that; they can be used in future works.

IV. RESULTS

Depending on the poll we did at the end of the semester, we see positive responses both from the students and instructors. Students expressed a strong preference for Codometer over classical way. Students also expressed that by using Codometer, they start to solve assignments more quickly (85.0% agree or strongly agree). Also students expressed that Codometer increases their confidence of their solution. In addition, they believed that provided excellent support for TDD (89.7% agree or strongly agree).

On the instructor side we see a 50% increase in time while preparing the homework, but the grading procedure has shortened by approximately 80%. So in overall, the spent time on an assignment decreases by 75%.
V. CONCLUSION AND FUTURE WORK

The experience in the classroom with Codometer, automated grading system, is very positive and increases the productivity of class. There can be seen a significant increase in the quality of student codes and decrease of human related errors. We plan to use Codometer in advanced programming courses this semester. In addition to this we are working on new features to Codometer such as instructor can set some measurements that emphasize the quality of code such code-comment ratio or cyclomatic complexity of solution. Another feature worth to add is cheat detection system, so that instructor can be notified if some submitted homework have similarities above some threshold value.

REFERENCES


