TOWARDS A PROGRESSIVE LEARNING ENVIRONMENT FOR PROGRAMMING COURSES

S. L. CHONG

Department of Computer Science, City University of Hong Kong
Tat Chee Ave, Kowloon Tong, Hong Kong

M. CHOY

Department of Computer Science, City University of Hong Kong
Tat Chee Ave, Kowloon Tong, Hong Kong

Learning with no feedback or delayed feedback slows down the learning pace and dilutes ones interest towards the subject matter. Grading programming assignment manually is a tedious and time-consuming process. This process involves repeatedly retrieve student programs, entering test data, examine the results generated, and then record and generate statistics for assessment evaluation. Under this assessment environment, feedback to students is often late and could not catch up with the learning pace. An automatic grading system is described in this paper aiming to provide an effective learning platform for tutors and students. Using such kind of system, it is believed that teaching staffs can shift their effort to the preparation and training instead of tedious work. In addition, more exercises can be made available for students to learn in a progressive manner.

1. Introduction

In recent years, the number of undergraduates studying computer science or related disciplines has been increasing rapidly. Programming is definitely one of the major skills required in these disciplines. Majority of the advanced computing courses presume that students are capable to master a range of programming languages such as C, C++, Java, etc. To ensure quality of programming skills, students need to engage a substantial amount of time to build up and master their skills. Practical exercises and assignments become an integral part of programming courses. In order to create a progressive learning platform, assignments should be graded carefully but quickly so that students can improve from useful and timely feedback. Consider the following learning cycle (Figure 1):

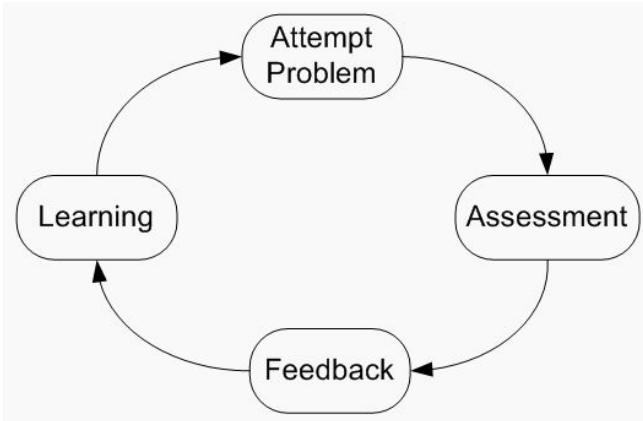

Figure 1. Learning cycle

Learning programming languages is similar to that of natural languages. One needs to be engaged in a lot of practices in order to master the tricks, the use of data structures, memory optimization, performance, etc. of a programming language. With the help of programming assignments, students are able to learn as well as get assessed on their skills level. After students attempted a program, tutor will mark the submitted work and feedback to the class. Through this assessment process, the tutor needs to spend time to inspect each program, check correctness and program logic, and then grade each student's work and write comment for them to get improved. The whole process involves a lot of routine work such as processing assignment submissions, compilation, execution and checking.

Whether these tasks are being done manually or semi-manually with the help of scripting tools, time spent in the assessment stage will be delayed by the degree of manual-ness in this stage [1]. This is a critical stage in the learning cycle. Any delay in this stage will slow down the whole learning cycle. Under normal circumstances this delay is certain, as tutors are facing a class of students. In addition to the class size, how the marking is being done will also affect the delay in feedback. How well the students can get improve from the feedback in the next learning cycle heavily depends on the time taken in this stage. This observation is particularly acute when the class size becomes increasingly large. In this way, the learning cycle gets slowed down further [1,3].

Knowing that learning is effective under a progressive learning environment where feedback is available, when we consider a fixed teaching

duration (normally 12-16 weeks), the number of learning cycles that can be achieved is limited in a progressive learning environment, therefore, students are forced to learn in a discrete learning environment (Figure 2) instead, where feedback of their work will not arrive on time for improvement in the next learning cycle (i.e. next assignment). To achieve a progressive learning environment, the learning cycle must be shortened. With short learning cycles, feedback should be made available to students rapidly, thus helping them to get improve in the next cycle. This study suggests an on-line submission and grading system for programming assignments that will speed up the marking process, and in return achieves a progressive learning environment with rapid learning cycles.

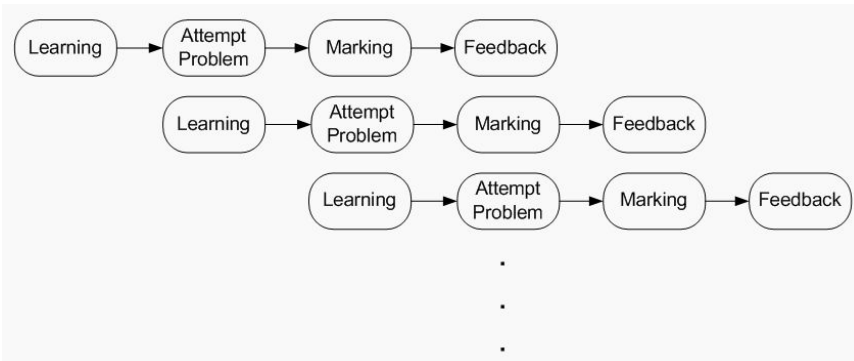

Figure 2. Overlapping learning cycles in discrete learning environment

2. System Overview

In the light of the above arguments, a web-based progressive learning environment is developed. Under this environment, the system is responsible for all the tedious tasks that were previously handled by tutors [2]. In a learning cycle, tutor posts assignment information together with some testing samples to the system. The class will be able to download the instructions and attempt the problem. Students can make use of the on-line sample test cases to test their work, or they can submit their programs to the system and let the system trial run for them. Students should submit their final programs to the system by the predefined deadline. Then, with the click of a button, the system will start the grading process, and the tutor will be able to read the assessment report from the system in less than a day. Assessment results and feedbacks can then be sent to students by the tutor. Writing of comments can be done by the system for automatically generation when test cases are carefully designed.

Aiming to replace manual or semi-manual grading process, this system eliminates the workload of teaching staff on repeatedly grading programming assignments. With this web-based progressive learning platform, it encourages students to explore their interests in programming, algorithms and related topics through short learning cycles provided by the use of this system.

3. Conceptual Model of the Grading System

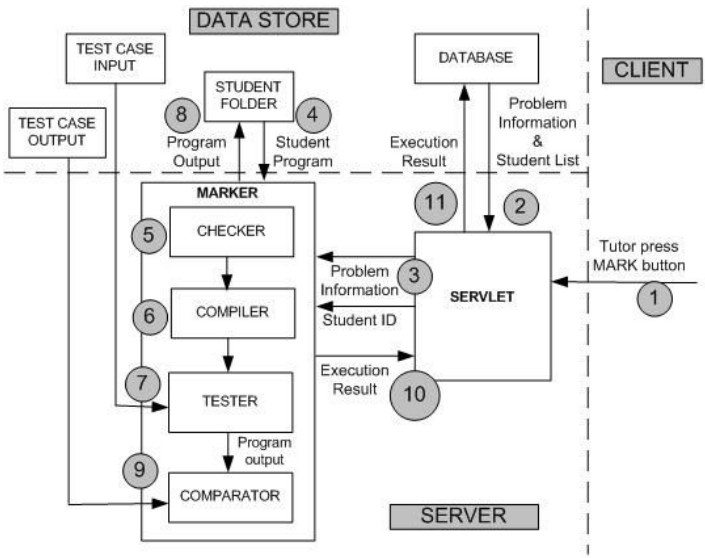

Figure 3. Conceptual model of programming assessment

Figure 3 describes a conceptual model of the program marking process. Assuming the deadline has past and students have submitted their programs online to the system. The grading steps used in this model are described below:

1. Tutor starts the grading process by pressing the "MARK" button, and control is passed to the system.
2. Servlet retrieves the list of students in this course and the problem information from the database.
3. Servlet sends all the relevant information to the system **Marker**.
4. System **Marker** retrieves a student program from the student folder.
5. System **Marker** sends the student program to the **Checker** to validate against harmful keywords.
6. If no harmful keyword resides in the program, system **Marker** will send the program to the **Compiler**. Java and Microsoft Visual Studio C/C++

compilers are available for tutor selection in this system. Other command line driven compilers can be included in the system as well.

7. If no compilation error occurs, the **Marker** sends the compiled program to the **Tester**. **Tester** then retrieves the set of predefined test cases, and feeds into the program to generate the corresponding outputs.
8. Outputs generated from the student program will then send back to the repository.
9. The **Comparator** then compares the set of generated outputs with the set of predefined outputs.
10. System **Marker** sends all correctness measures (such as number of completed test cases) to the servlet.
11. Servlet sends the execution result to the database for recording.

System Marker will continue to grade the next student program by repeating above steps until the set of student programs are graded.

4. Practical Issues

In the on-line grading sub-system, compilation and execution of programs are unavoidable operations. Some programs may, by design or accident, contain code that adversely affects the system. In response to this, preventive actions should be done. First, file with extension other than those specified in the assignment problem specification will be rejected. Second, all uploaded files will be scanned once before compilation against a predefined list of harmful words.

Besides, queue hogging will be happened if students use this system as a testing tool or some students repeatedly submit programs to the system. In response to this, the tutor can set a limit to the maximum number of submissions and another limit to the queuing up for trial run.

Moreover, to enhance the automatic ability of the system, a scheduler is to be implemented so that after the submission deadline, the system Marker will start automatically according to the predefined schedule. In addition, plagiarism checker and test case generator are recommended to this system.

5. System Reliability

The system is being tested thoroughly using unit test, integration test and stress test. The testing results are encouraging, no critical problems have been encountered during the test. When the system was demonstrated to a number of tutors, they found this system user-friendly and helpful in their teaching. A pilot run was scheduled towards the end of spring semester this year for one of the programming courses in our department. This system automates the

submission of programming assignment as well as the marking of correctness of these student programs.

6. Benefits from this System

In this paper, a web-based submission and automatic grading system is described, aiming to build up a progressive learning environment to assist the learning of computer programming. The major achievements are summarized as follows:

- This system provides teaching staff an effective and fast measure in the marking process.
- Through the use of this system, time spent in tedious work is reduced. The system provides a channel for feedback on student programming assignment, as well as generates common statistic of the class performance. In addition, the learning cycle is shortened. Students are able to improve from the feedback in their next assignment.
- The web-based platform provides students with an easy-to-access channel to submit and check their assignments, and a testing environment with sample test cases.
- This system is suitable for any programming language having a command line driven compiler.
- A progressive learning environment is achieved within the teaching period, as more rapid learning cycles are made feasible. The improvement in the quality of student learning will then be the ultimate benefit from this system.

References

1. A. Kurnia, A. Lim and B. Cheang, *Comp. Ed.* **36**, 299 (2001).
2. D. Jackson, *Comp. Ed.* **27**, 171 (1996).
3. P. C. Isaacson and T. A. Scott, *ACM SIGSCE* **21**, 15 (1989).