Automated Assessment for Improving the Learning of Computer Programming: Potentials and Challenges

Usman Nazir, C. K. Poon, Y. T. Yu and Marian Choy

Department of Computer Science, City University of Hong Kong

{usmann, csckpoon, csytyu, csmchoy}@cityu.edu.hk

Abstract: *This paper presents a Web-based automatic programming assignment assessment system called PASS. The system is designed to overcome the tedious and time-consuming task of manually grading student assignments in programming courses. The paper discusses how automated systems, such as PASS, can be used to facilitate the process of grading student assignments. It gives a brief description of the system together with some of the practical issues handled in its development. PASS has been successfully used for two programming courses using C and C++ languages, with encouraging feedback from students. The system is currently being enhanced to provide more functionalities and is planned to be used in other computer programming courses as well.*

Keywords: Automatic grading system, online programming assessment system, PASS

1. Introduction

The popularity of computer programming courses has necessitated the automation of the routine and tedious manual tasks to cater for the growing number of students taking up these courses. One such aspect that needs to be facilitated through automation is the time-consuming process of grading students' assignments.

The process of submission and grading of assignments in a programming course is basically divided into the interaction between two parties: students (programmers) and tutors (markers). A student programs the assignment and submits it to the tutor for grading. The tutor grades the assignment and gives feedback to the student on his/her work. The student can then use the feedback to enhance his/her programming skills. Although the process is practical and simple, it usually takes a long time for a tutor to grade the assignment and return feedback to the student for his/her learning. This problem is further aggravated by the growing number of students taking up computer programming courses. The more students there are the longer it will take to grade the assignments and return feedback to the students. Programming is a skill that is acquired through practice (Kurnia et al., 2001). The more a student practises, the better he/she gets at it. Assignments help students practise their programming skills and therefore it is important that they can receive timely feedback on their work that can benefit their progress.

Automating the grading process can help reduce the considerable amount of time and routine work invested by the tutor in grading. This in turn also means that the tutor can design more practice assignments for students to enhance their programming skills. PASS, presented in this paper, is one such system that has been successfully used in City University of Hong Kong to automate the programming assignment submission and grading process for two computer programming courses: CS2362 (which uses C Language) and CS2331 (which uses C++ Language).

The sections that follow in this paper firstly discuss the problems involved in the traditional way of grading assignments and how automated systems such as PASS can overcome these problems in facilitating the grading process. The system PASS is then described, followed by the achievements and challenges of using the system, a brief comparison with similar work, and conclusion.

2. Problems in Manual Grading

Manual grading has traditionally been the usual practice for grading programming assignments. Students are required to submit their program source codes to their tutors before the deadline for marking. Marking is a complex process in which the markers are required to examine the source codes thoroughly and in many cases compile and run the programs with some predefined test cases. The markers are also required to check whether the programs can produce some specified outputs with given input parameters. In the end the marker has to give comments and feedback to each student according to their performance (Ng & Ngai, 2003).

Manually grading all the student assignments can be an inefficient, tedious and error-prone process. Usually a marker has to work fast to meet the tight deadlines. No matter how carefully designed the marking scheme is, there is always a possibility of human errors that may result in erroneous grading. Employing several markers may speed up the process, but it may create a problem of different markers having their own marking styles and preferences, which may affect the grade that a student gets. The same program when graded by different markers will frequently result in different grades. In fact, the same person may give a different grade to the same program depending on his mood, alertness or different factors (Cheang et al., 2003).

When the marker gets the program files submitted by the students, even in a first programming course, it is a difficult and complex process for a marker to assess the program correctness accurately and comprehensively simply by reading the source code (Kay et al., 1994). The marker usually has to set up a testing environment, compile and run student programs using a predefined set of inputs. Compiling and running hundreds of programs can be a hassle. This considerably increases the workload of the marker and as a result factors such as stress and mood shifts may play a part in the grading process. Owing to the large amount of work a marker has to do in marking the assignments, he/she may be reluctant to re-mark the assignments even if a student deserves a better grade (Kurnia et al., 2001). The environment set up to mark a student's assignment may be different from what the student used to test his program. Therefore, even if the program produces correct results on the student's system, he/she may end up losing marks because his/her program does not do as well on the marker's system. The marker's job may be made easier by asking students to provide evidence of their own testing, but this requires further skills on the part of the student that may not be covered in the course. Moreover, students may alter their true test results to be submitted in order to get a better score in the assignment (Luck & Joy, 1999).

Another serious problem may arise if the tutor obliviously runs a malicious program submitted by a student on his system. This may result in the loss of all the data on the system, perhaps together with all the assignments submitted by the students (Reek, 1989). Securing the system and making backups before testing the assignments is an issue that all markers must take care of.

As the marker has to or should take care of every single aspect of assessing the student assignment correctly and fairly for each and every student in class, it can be a very time-consuming process which is prone to human errors. This problem is aggravated by the growing size of classes due to the increasing number of students taking up programming courses.

3. Some Benefits of Automated Assessment

Automated systems are always consistent in their behaviour; they give the same result every time they are run. This makes them ideal for assessing assignments where fairness and consistency is highly desired. They can work without rest for long hours and do not suffer from factors such as mood swings.

Automated systems can be deployed to perform all the tedious and routine tasks such as locating, storing and organising the student programs. They can generate and compare the outputs for a match and perform all the calculations that are needed to give the final score to a student for his/her work. Automating such tasks relieves tutors' burden and save them time, which they can use to design ways to help their students better understand the concepts taught. All the assignments can be marked in a relatively small amount of time. This helps to provide early feedback to students for their learning, and more and smaller assignments for students to practise their programming skills. The learning process becomes more interactive: Instead of using one large assignment to test the students on many aspects, the tutor can design a number of shorter exercises and students can learn from the timely feedback much more quickly than before.

The system which is used for marking can be made available to the students for testing. A student can test his/her assignment so that many mistakes can be corrected before submission. This practice greatly increases students' confidence in their own work and helps them achieve better results through a more interactive learning environment.

4. PASS: Programming Assignment aSsessment System

4.1. System Overview

PASS is an automated Programming Assignment aSsessment System that is being used at Department of Computer Science in City University of Hong Kong. The prototype version of PASS was initially built for demonstration purposes (Chong & Choy, 2004). It has now been fully developed, tested and deployed into production use in two computer programming courses.

Through PASS, the tutor may upload the assignment information for his course together with some public test data for use by students. Students can obtain the problem specification from the system and test their programs online using the public test cases provided by the tutor for the assignments. Testing their assignments online gives them instant feedback on their work so that they can improve and retest their programs as many times as necessary. Students can submit their programs to the system for marking before the deadline specified by the tutor. To start the grading process, all the tutor has to do is to click a button and he/she will be able to receive the assessment report in about a day. Students can get to know the results of their assignments very soon after they are marked, together with the feedback manually added by the tutor. Figure 1 summarises the interaction between the tutor and the student through the system.

PASS relieves the workload of the tutor in marking the assignments and saves his/her time that might better be spent elsewhere, such as coming up with ways to enhance student understanding of the subject. Students can use PASS to test their assignments with the public test cases provided by the tutor and get instant feedback from the system on the work submitted. The tutor has to provide a set of private test cases designed specifically for assessing students' programs. It is important to keep this test set private. Otherwise, a student may just tailor-make his/her program to print the required output of the public test set without solving the real problem.

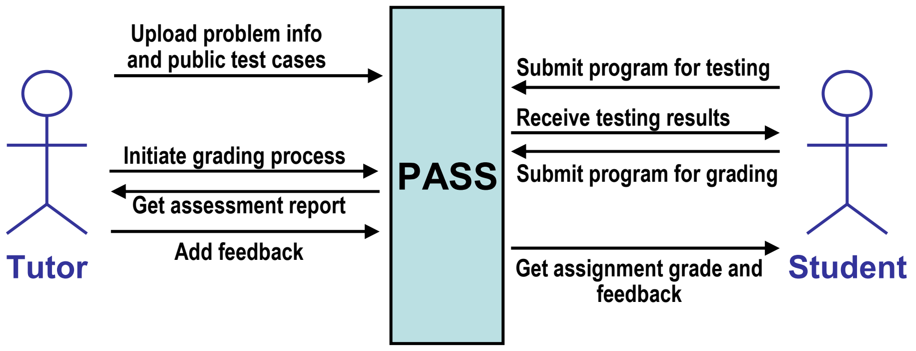
Upload problem info and public test cases
Initiate grading process
Get assessment report
Add feedback
Tutor
PASS
Submit program for testing
Receive testing results
Submit program for grading
Get assignment grade and feedback
Student

Figure 1. Interaction between the tutor and the student through PASS

4.2. The Grading Process

Figure 2 depicts how the grading process is carried out in PASS. After the deadline has passed and students have submitted their programs online to the system, the steps involved in grading an assignment are as follows.

1. Tutor initiates the grading process, and control is passed to the system.
2. Servlet retrieves the list of students of the specified course and problem information from Database.
3. Servlet sends all the relevant information to Security Module.
4. Security Module retrieves a student program from Student Folder.
5. Security Module sends the student program to Security Filter for validation against harmful keywords.
6. If no harmful keywords in the program are detected, Security Module will copy the student program and place it in a secure location called Execution Folder, where running any kind of executable file through PASS will not harm the system.

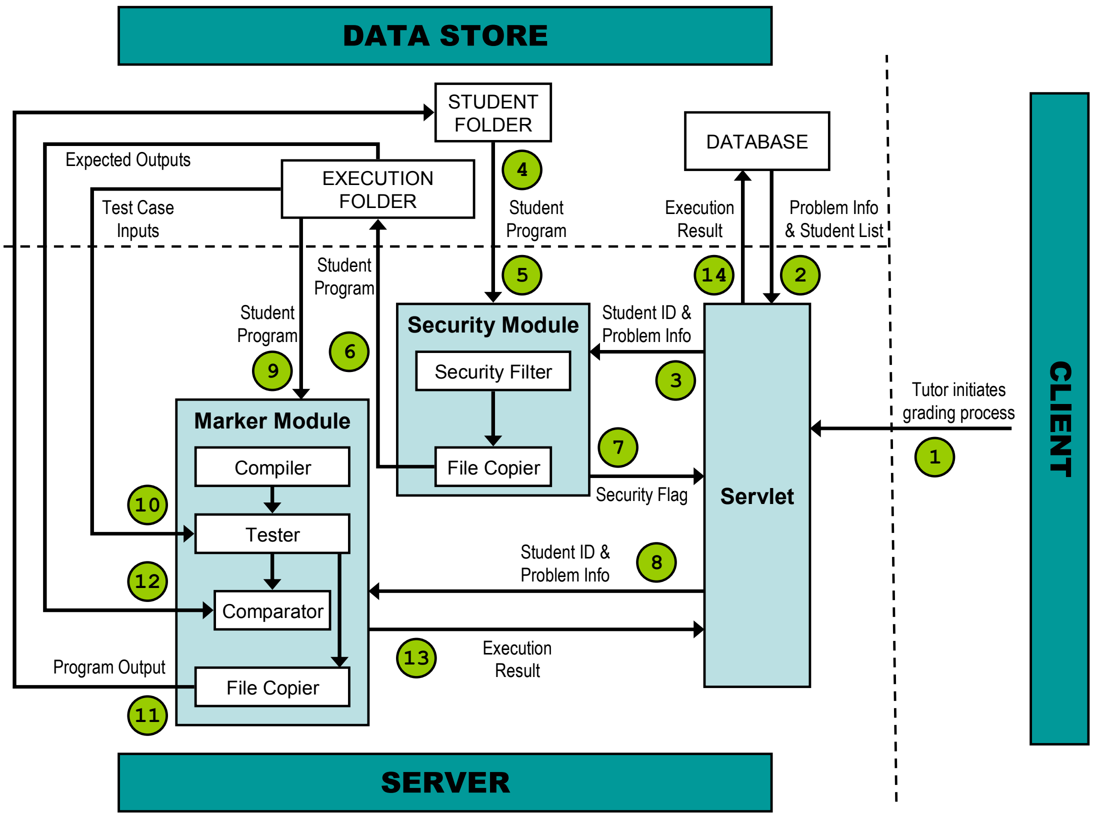

Figure 2. The grading process in PASS

7. A Security Flag is sent back to Servlet indicating whether it is safe to run this program or not.
8. If it is safe to run the program, Servlet sends all the relevant information to Marker Module.
9. Marker Module retrieves the student program from Execution Folder and compiles the submitted source files via Compiler.
10. If no compilation error occurs, Marker Module sends the compiled program for testing via Tester. Tester retrieves the set of predefined private test cases, feeds them into the student program for execution to generate the corresponding outputs, and stores the outputs in Execution Folder.
11. Outputs generated from the student program are copied to Student Folder.
12. Comparator compares the set of generated outputs with the set of predefined expected outputs.
13. Marker Module sends all correctness measures (such as the number of successfully executed test cases) to Servlet.
14. Servlet sends the execution result to Database for storage.

Afterwards, Marker Module then continues to grade the next student program by repeating the above steps until all student programs are graded.

4.3. Securing the System from the Malicious Programs

It is important that the system be secure. We have used two approaches of protecting the system from being harmed by malicious programs.

(1) *Scan the source file for harmful keywords*

Initially, PASS was designed with this security feature. Every source file submitted by students was scanned for suspicious keywords that may harm the system. Although this approach of detecting malicious programs is a practical one and initially rather effective, it would be easy for a hacker to find ways to get around with this kind of protection once the mechanism is known. Also, the system administrator has to constantly keep the system up-to-date with all the possible keywords that may cause harm to the system, and it will take great effort to extend the system to other programming languages. Although PASS does not rely on this feature any more, it is still kept available as an option for situations in which the tutor would like to disallow any particular keyword in students' submissions.

(2) *Execute the potentially malicious program in a relatively safe environment*

PASS uses MS Windows Server 2003 Operating System both in the production and the development environment. MS Windows Server 2003 provides a feature where the system administrator can run executables under the guise of a different user account. A guest account with restricted capability to access system resources is created for execution of student programs. The guest account is only allowed to access one directory, namely, Execution Folder in the system. Students' programs are copied to Execution Folder and executed under the guest account. This provides very good protection of system files and resources against malicious programs.

PASS also offers protection at the user level where only the types of the source files needed for the course language are allowed to be uploaded and all other file types are rejected.

4.4. Protection Against Too Many Users Trying to Execute Their Programs at the Same Time

If too many users run their programs at the same time, the system resources will eventually be exhausted. System performance will become too slow, and PASS may run out of memory while executing memory-intensive programs. To prevent the adverse effects of too many programs running at the same time, PASS imposes a limit on the number of concurrently active programs in execution. Currently the prescribed limit is set to 10. This facility is implemented by using a waiting queue. When 10 programs are already running in PASS, any new requests to run a program in the system would have to wait for their turn, until some of the running programs have terminated. This feature has proven to be very reliable and does offer more control over managing logs and tracking the errors.

4.5. Logging the User Activity and System Exceptions

Logging user activities sounds like a simple task, but if improperly done, it may considerably slow down the system performance. PASS employs the use of the utility provided by Apache called 'log4j' to log user activities and exceptions in the system. The use of log4j has proven to be very efficient and appropriate for PASS.

4.6. Kill the Programs If They Take More Than the Specified Running Time

When uploading a problem, the tutor supplies the maximum time allowed for a program to run on the system. Programs that execute longer than this specified time limit are considered a threat to the system and will be killed.

However, PASS does not use elapsed system time for this purpose, because usually many programs are running on the system at the same time and they are scheduled differently by the Operating System to be invoked at different times depending on the load of the system. Instead, PASS tracks the CPU time actually used by the program. This is a fairer and more accurate way of controlling the execution time of a program. To do this, an open source utility called 'pslist' is used to monitor the execution of programs. A program is killed as soon as its execution time exceeds the specified limit.

5. Accomplishments and Challenges

PASS has been used for two computer programming courses at Department of Computer Science in City University of Hong Kong during the Fall semester in 2004. The course CS2362 used C language and about 280 students used PASS. The course CS2331 used C++ language and about 110 students used PASS. Students of these courses have been using PASS not only to test and submit their assignments, but also for their lab exercises. The tutors of these courses used PASS to upload problems, test the uploaded problems and mark the student programs.

5.1. Reliability and User Acceptance

Altogether, about 400 students have used PASS in their courses. So far, PASS has proven to be resilient enough for the tasks, and the system performance has been very good. It did not show any major problems in its functionality. Some minor bugs in the programs had been discovered but they were fixed unnoticeably quickly. The system administrators of PASS did not receive many complaints and the tutors of the courses did appreciate the use of PASS in their courses. The tutors who used PASS in their courses have indicated their wish to use it again in the upcoming semester.

5.2. Security

An automated system for reporting anomalies by students has been in place soon after PASS was deployed for production, and so far no security breaches have been reported. PASS has provided good file system security such that no student can copy, update or delete other students' programs. Students can check their own uploaded files any time on the system, providing them assurance that their submitted programs are being processed.

5.3. Statistical Analysis of the Survey

A survey was conducted in the CS2331 course for evaluating PASS. Students were handed a questionnaire asking for their opinions and suggestions. The majority of the students reported that they used PASS quite often and agreed that PASS had been very helpful for the lab exercises. They reported that on average they tested their assignments 5 to 10 times before submission. Many students suggested to relax the strict format of the outputs required by PASS for checking their correctness. Table 1 summarises the results of students' rating of PASS.

Table 1. Students' rating of PASS

Question: On the scale of 1 to 5, how would you rate PASS?
(5 being the highest / best rating)

Rating	Number of Students
5	13
4	52
3	14
2	6
1	1
Total	86

The feedback and responses obtained from all users have been very encouraging. Most students agreed that PASS had been very helpful in facilitating their learning process. All the tutors did appreciate PASS, found it very helpful and efficient in marking student assignments and would like it to be available for the next semester.

5.4. Extensions

Although MS Windows 2003 Operating System does show good resource management, it will be desirable to programmatically control the memory usage by each program. Planned extensions to PASS include plagiarism detection and automatic handling of feedback to students. PASS is also planned to be used in courses that use other programming languages.

6. Related Work

Many researchers and institutions have realised the need for automated systems for assessing programming assignments, and similar work has been implemented to reap the benefits of such a system. Reek (1989) developed a UNIX based system called TRY that is similar to PASS in functionalities, except that PASS is an online web-based system that uses Microsoft Windows Server 2003 as its platform, does not require a user to put their programs in a specified directory structure and places a time limit restriction on the programs that are executed on the system.

Jackson (1996) describes another system that uses UNIX as its platform. In addition to checking correctness and efficiency, his system also checks the style and complexity of student programs. Luck and Joy (1999) developed a UNIX based system called BOSS. Both BOSS and PASS allow online submission, testing and marking of coursework. The security aspect of BOSS is also well handled, that is, it provides facilities to ensure data integrity, provides data privacy and protects the system against rogue programs. BOSS is relatively more comprehensive and provides more functionalities, such as plagiarism detection, mark sheets and moderation sheets. However, the use of MS Windows Server 2003 in PASS makes it easier to use popular C/C++ compilers that run in the MS Windows environment, with which our students are more familiar.

7. Conclusion

This paper has described a web-based automatic programming assignment assessment system called PASS that provides an environment for facilitating the learning of computer programming. It has relieved the workload of tutors in grading student assignments. Students can use it to test and submit their assignments online. PASS has also enabled the tutor to return timely feedback to students on their work, which helps to enhance their programming skills.

PASS has been successfully used in two computer programming courses during the Fall semester in 2004. Responses obtained from the users have been encouraging, and PASS is planned to be used in other computer programming courses as well in the coming semester.

8. Acknowledgments

The work described in this paper was partially supported by Teaching Development Fund (project no. 6980041) from City University of Hong Kong.

References

Cheang, B., Kurnia, A., Lim, A., and Oon, W.-C. (2003). Automated Grading of Programming Assignments. *Computers & Education* 41(2), 121-131.

Chong, S.L., and Choy, M. (2004). Towards a Progressive Learning Environment for Programming Courses. In Cheung R., Lau R., and Li. Q., (editors). *New Horizon in Web-based Learning: Proceedings of the 3rd International Conference on Web-based Learning (ICWL 2004),* World Scientific Publishing Co. Pte. Ltd., pages 200-205.

Jackson, D. (1996). A Software System for Grading Student Computer Programs. *Computers & Education,* 27(3/4), 171-180.

Kay, D.G., Scott, T., Isaacson P., and Reek, K. A. (1994). Automated Grading Assistance for Student Programs. In *Proceedings of the 25th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'94),* ACM Press, pages 381-382.

Kurnia, A., Lim, A., and Cheang, B. (2001). Online Judge. *Computers & Education* 36(4), 299-315.

Luck, M., and Joy, M. (1999). A Secure On-line Submission System. *Software Practice and Experience* 29(8), 721-740.

Ng, S.C., Li, T.S., and Ngai, H.S. (2003). An Integrated Assessment System for Marking Programming Assignments. In *Proceedings of International Conference on Computers in Education (ICCE 2003)*, pages 699-703.

Reek, K. A. (1989). The TRY System -or- How to Avoid Testing Student Programs. In *Proceedings of the 20th SIGCSE Technical Symposium on Computer Science Education (SIGCSE'89),* ACM Press, pages 112-116.