Submit! A Web-Based System for Providing Automatic Feedback to Large Classes

Yusuf Pisan, Debbie Richards, Anthony Sloane

Computing Department, Macquarie University, Sydney, Australia

ABSTRACT

This paper[1] presents the Submit! project which aims to enhance teaching and learning in computing by developing automated web-based tools that assist in providing critical feedback to students about the computer programs they write. By developing sophisticated computer-based tools that will improve our monitoring of student progress and maintenance of consistent standards we aim to provide structured assessment with a level of detail and consistency that would be difficult or impossible to provide manually. By allowing students to use the critiquing tools before final submission of an assignment we offer formative assessment that supports self-directed learning. Submit! has been integrated into many computing units at Macquarie University. Usability evaluations show that Submit! generally effective while needing improvement in certain areas. A preliminary study of the impact of Submit! on student results shows that students who make use of the system to get feedback on assignment submissions do better than those who don't.

KEYWORDS

integrated learning environments, code reviews, assessment and feedback

INTRODUCTION

This paper presents a project to enhance teaching and learning in computing by developing automated web-based tools that assist in providing critical feedback to students about the computer programs they write.

By developing sophisticated computer-based tools that will improve our monitoring of student progress and maintenance of consistent standards we aim to provide structured assessment with a level of detail and consistency that would be difficult or impossible to provide manually. By

[1] A longer version of this paper containing much more detail regarding the evaluation studies conducted has been submitted to the *5th Australasian Computing Education Conference (ACE2003)* to be held in Adelaide.

allowing students to use the critiquing tools before final submission of an assignment we offer formative assessment that supports self-directed learning. The tools, collectively known as the *Submit!* system, are becoming an important part of the on-line support for our units, providing a means of giving students the same quality of assessment regardless of the delivery mode or class size.

In this paper we first provide the background and rationale of the project. The next section introduces two usability studies, one from the student and another from the academic perspective, and an examination of the effects of the system on student performance on a first year assignment. Our conclusions and future work are given in the final section.

BACKGROUND AND APPROACH

Most computing units use programming tasks to reinforce lecture and tutorial material with practical experience. *Program critiquing* refers to the process by which students obtain critical feedback about their programs. Critiquing can be used as a component of formative or summative assessment to enhance self-directed learning. In fact, providing timely and high-quality feedback has been identified as one of the critical factors in learning (Ramsden, 1988; Gibbs, 1994; Hattie, Jaeger and Bond, 1999).

The current and traditional method of providing critiquing is to do it manually: tutors or lecturers read submissions, optionally run programs on test data, and make comments. In large computing units, such as those in first year where there can be close to one thousand students enrolled in a unit, the amount of time required for critiquing makes it prohibitive to give feedback to students on more than two or three assignments per semester. As a result, students have limited opportunities to learn from their mistakes, and it is difficult for lecturers to monitor student progress.

The alternative to manual critiquing is to automate the process. Current commercial systems are restricted to true/false or multiple-choice questions and are of limited use for deeper learning. A number of intelligent learning environments have been developed as research prototypes. Some well-known examples are SOPHIE (Brown et al., 1982), LISP Tutor (Anderson and Reiser, 1985), Geometry Tutor (Anderson et al., 1986), Sherlock (Lajoie and Lesgold, 1989), WebToTest (Arnow and Barshay, 1999), and PILOT (Bridgeman et al., 2000). Much can be learned from these systems in terms of underlying techniques; however, they are limited to specific domains and, most importantly, do not address the program critiquing problem.

Although a variety of testing and critiquing programs had been developed within the department, these programs have lacked a uniform interface, been difficult to use by people other than the original author, and have often duplicated previous efforts.

The goals of the Submit! project are:

- to develop a suite of tools that specifically support program critiquing;
- to provide a uniform interface to these tools; and
- to ensure that the tools are designed with usability in mind.

The process of critiquing programs and generating high-quality feedback is a complex task. Despite this complexity, program critiquing is feasible because each programming language has a well-defined syntax and semantics that makes the task of automation easier than it would be for free-form natural language submissions such as essays or reports.

EDUCATIONAL RATIONALE AND CONDUITS TO STUDENT LEARNING

We expect automated program critiquing to deliver two major benefits to students: improved feedback and more useful access to lecturers.

Feedback will be better because:

- It will be specific to the individual's submission rather than a general list of possible errors or solutions that the student is often unable to apply to his or her own case.
- There will be more feedback as comments can be added every time the need is identified, in contrast to the current circumstance where the quality of the commenting varies as a result of marker fatigue and time pressure. The feedback will be more consistent and objective as the computer will always provide the same detailed feedback regardless of whether it is the first or the last student who has made the mistake.
- The feedback will be more timely because the computer will be able to process solutions quicker than a human.
- Formative feedback will be possible because in some cases students will be able to run the critiquer themselves before final submission to allow them to correct their own mistakes. There is more incentive for students to spend the time understanding what they did wrong when it can have a direct impact on their marks. Feedback offered after marking is often not incorporated or even read, as evidenced by the number of assignments that never get picked up. Also, students often complain that they did not understand what was expected of them. Being able to test out a solution before submission should reduce this problem. Advanced students will be able to proceed to the next set of exercises and get feedback on their progress instead of being forced to wait while the topic is covered in lectures.

Access to lecturers will be more useful because:

- The more mundane questions such as "What did I do wrong?" will be answered by the critiquer. This means that lecturer-student interaction can concentrate on the concepts underlying the solution.
- Freeing lecturers from some of the marking task will result in an increase in the time available for consultation with students.
- Students and lecturers should find interaction more satisfying as the nature of the discussions should be more stimulating and less shallow.

The project will also provide substantial benefits to our department:

- The department has an ongoing plan for quality improvement throughout its curriculum, based on standardisation and documentation of procedures across all units. By assisting in the enforcement of consistency in assessment, the project addresses an issue that significantly affects the quality of teaching and learning in Computing at Macquarie.
- The project will lead to a reduction in our reliance on casual staff and the consequent costs. Some casual staff assistance will always be necessary to assist in assessment, but as students take a more self-directed learning approach in computing units, these staff will be able to focus on more productive assistance rather than routine assignment marking.
- The generality of the system we are developing means that a broad range of computing units will benefit as a result. We are not focussing on one particular programming language, since the Department of Computing uses a variety of programming languages, and the range used changes over time. Our general approach and system will thus support a wide range of units as well as allow the department to keep up with the latest trends in computer programming languages.

MONITORING AND EVALUATION

Evaluation is a key aspect of all phases of the Submit! Project. Our evaluation of the tools developed is based on four criteria:

1. the usability of the tools;
2. the attitudes of both staff and students towards the tools;
3. the impact of the tools on students' scores in programming tasks; and
4. the impact of the tools on students' deeper knowledge of course content.

To date we have conducted evaluations to assess the first three criteria: two usability studies, one from the student perspective and one from the staff perspective and an initial evaluation of the impact of the tool on student scores.

The aim of the student usability study was to identify the strengths, weaknesses and possible improvements that would make Submit! a more user-friendly system for students to use. In identifying the strengths and weaknesses we were able to establish the level of intuitiveness, that is, whether a student could easily figure out the steps to complete a task without assistance by persons or by help. Finally, we were able to identify some useful features whose implementation will improve Submit!. The evaluation consisted of heuristic evaluation, observation of students performing given tasks and a questionnaire.

The academic usability study focused on the aspects of the system used by markers and lecturers. These users have entirely different objectives compared to students. Their main concern can range from simply downloading a zip file of students assignments for marking, to the administration of assignments and subjects, whereas students are mainly concerned with simple submissions. A number of instruments were used in this study including: surveys, a cognitive walkthrough, heuristic evaluation and comparison with "rival" systems from other universities.

Tasks that were assessed from the academic perspective included:

- Basic Assignment Creation
- Advanced Assignment Creation -this includes the restriction of submissions by file name, by file size, and by enrolment, selected students and date.
- Modification of the Assignment Parameters
- Automatic Testing of Submissions
- Reporting – allowing lecturers to observe a listing of students and their results for the current assignment, sortable by result and other criteria such as first / family name.

The Cognitive Walkthrough is a think-aloud method for assessing the user's mental model and problems they are encountering in performing specified tasks. The process involved taking a user through a series of eight tasks and simulating the users problem-solving process at each step in the users "dialog" with the system. For each of the tasks the investigator asked:

- Was the correct action sufficiently evident to the participant?
- Was the correct action noticed when it was available? For example, was the help button visible when needed?
- Did the user know from the feedback that actions where successful?

In short, from the academic perspective, Submit! performed very well in the cognitive walkthrough and participants gave it better than neutral ratings for most of the Likert statements in the questionnaire. As a result of the questionnaire and the Heuristic Analysis, many suggestions have been made and usability problems identified.

EVALUATION OF IMPACT OF SUBMIT! ON STUDENT RESULTS

We have conducted an initial study in one large first year programming unit. The study uses data over two semesters: S1 2002 and S2 2002. In Semester 2 we have incorporated programming style checking scripts into Submit!. Students are able to submit their assignments as many times as they wish prior to the assignment due date and receive feedback on their programming style. The guidelines were chosen based on our perception of most common problems in student code and the ease of implementing the automated tests.

In Semester 1 students used Submit! to submit their assignments, but could not run the automated tests to get feedback on their style problems. This semester we have made automated testing available. At this stage we have only collected data for assignment one, shown in Table 1. The students show slight improvement in most of the categories, except for the "too long" category. One possible explanation for this is that unlike the assignment in Semester 1, the assignment in Semester 2 required students to print multiple lines of text.

Criteria	S1 no feedback	S2 feedback
Number of Assignments	232	261
Too Long	29%	31%
Tabs	92%	14%
Paren_Curly	24%	14%
Code_Curly	24%	13%
Block_Single	22%	8%
Comment Space	10%	7%

Table 1: Percentage of students who had a style error with and without feedback prior to final submission

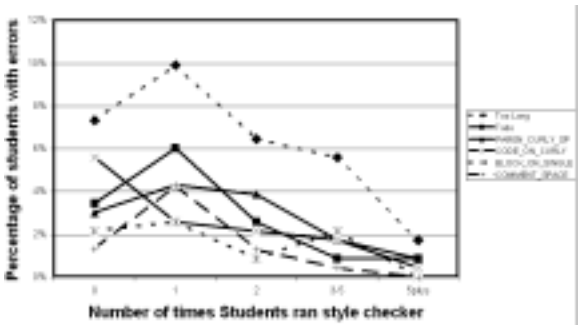

Figure 1: Error trends based on how many times students ran style checker

We found that only 77% of the students ran the automated test. We attribute this to either students not being aware of this new facility or to students submitting the assignments right before the deadline leaving no time to fix any mistakes. There was no limit on how many times students could run the automated style checker. We expected that some students would eliminate all their errors by repeatedly running the style checker. Some students ran the checker over 10 times. We were interested to see if there was any trend on the amount of usage and the assignment results. To check this we grouped the data according to usage categories (once, 2-5 times, more than 5 times) and analysed the number of errors for each group.

Figure 1 shows the general trend that students who ran the style checker more times have fewer errors. We noticed that students who ran the test once had slightly higher error rates on a number of indicators. There are two possible explanations for this effect:

1. The style checker takes anywhere from two to ten minutes to complete. For students who ran the style checker right before the assignment deadline, by the time the style checker was complete the assignment would already be locked preventing students from fixing their mistakes.
2. In addition to the style checker, there was also a basic input-output checker that was available to the students. Students who ran both the input-output test and style checker may have decided to fix mistakes related to input-output rather than style errors. We are currently trying to test these hypotheses by talking to students and examining system logs.

CONCLUSION AND FUTURE WORK

So far the Submit! project has met its goals in providing a consistent and effective framework in which feedback can be given to students on their programming work. Our incremental approach has enabled even early versions of the system to help students and teachers get their work done more effectively.

Our evaluations have shown that the system is generally usable while pin-pointing areas of improvement that we are addressing in current development. Moreover, a preliminary comparative study of student assignment performance has shown that providing feedback for self-assessment can provide benefits.

In the future, we would like to capture and encode the knowledge we gain about useful forms of critiquing so that it can be reused. In particular, we intend to create a library of reusable modules to enable similar feedback for programs written in different programming languages. In developing the libraries, we will discuss with students and staff how feedback should be structured such that the feedback is informative and thought-provoking for the students without making them reliant on the system.

We will also incorporate plagiarism detection to check for similarities between submissions and notify staff of any suspicious cases.

As a longer term goal, we will explore how we can support critiquing for other types of submissions. For example, in some of our units students are required to submit designs rather than actual program code. The designs are expressed using formal graphical notations, so we should be able to convert them to text for analysis. Any discipline that makes similar use of structured submissions will be able to adapt our tools for use in their units

ACKNOWLEDGEMENTS

We are grateful to Macquarie University for funding to support this project under the Macquarie University Flagship Teaching Development Grant scheme. We also acknowledge the significant assistance and feedback provided by academic, student, administrative and technical support members of the department and division.

REFERENCES

Anderson, J. R., And Reiser, B. J. (1985) The Lisp Tutor. *Byte*, 10, Pages 159–175.

Anderson, J. R., Boyle, C. F., And Yost, G. (1986) The Geometry Tutor. *The Journal Of Mathematical Behavior*, Pages 5–20.

Arnow, D. And Barshay, O. (1999) On-Line Programming Examinations Using Webtoteach. In *Proceedings Of Innovation And Technology In Computer Science Education*, Cracow, Poland, Pages 21–24.

Bridgeman, S., Goodrich, M. T., Kobourov, S. G. And Tamassia, R. (2000) Pilot: An Interactive Tool For Learning And Grading. In *Proceedings Of The 31st Acm Sigcse Technical Symposium*, Austin, Texas, Pages 139–143.

Brown, J. S., Burton, R. R., And Dekleer, J. (1982) Pedagogical, Natural Language And Knowledge Engineering Techniques In Sophie I, Ii And Iii. Pages 227–282 In D. Sleeman And J. S. Brown (Eds.), *Intelligent Tutoring Systems*. New York: Academic Press.

Gibbs, G. (Ed) (1994) *Improving Student Learning — Theory And Practice*. Oxford: Oxford Centre For Staff Development.

Hattie, J.A., Jaeger, R.M., And Bond, L. (1999) Pervasive Problems In Educational Measurement. *Review Of Research In Education*.

Lajoie S.P. And Lesgold A. (1989) Apprenticeship Training In The Workplace: Computer-Coached Practice Environment As A New Form Of Apprenticeship. *Machine Mediated Learning*, 3(1), Pages 7–28.
Ramsden, P. (Ed) (1988) *Improving Learning: New Perspectives*. London: Kogan Page.