Programming students NEED instant feedback!

Anne Venables and Liz Haywood

School of Communications and Informatics
Victoria University of Technology
PO Box 14228, Melbourne CMC, 8001, Victoria

[anne.venables, liz.haywood]@vu.edu.au

Abstract

This paper describes the work that has been undertaken to improve the quantity and quality of feedback given to students studying introductory Java programming. An on-line submission program (submit) was developed that specifically addresses the needs of beginning programming students by providing automatic and instant feedback to a student's programming efforts. Traditionally, this feedback was provided by staff in scheduled laboratory sessions, yet the amount of individual attention a particular student received was haphazard. In addition, the introduction of the submit program has reduced the marking and administrative load associated with providing timely feedback to students for participating staff.

submit is more than just an on-line letter box for students to post their programs. submit also provides instant feedback which encourages students to improve their programs to meet all given requirements as well as to improve programming style.

In order to assess the effectiveness of the submit program over four trials, student responses to questionnaires were collated and support some tentative conclusions as to the effectiveness of submit as a learning aid to students. Staff opinion from those involved in the day-to-day teaching of Java programming was also solicited. These responses have been very positive, especially from the staff that supervise laboratory work.

Keywords: Computer aided learning, on-line submission.

1 Introduction

The Australian Computer Society (ACS) considers program design and implementation to be a core area of knowledge required by a professional in the computer industry (ACS 2002). At Victoria University of Technology, Java is taught as the introductory programming language to all students studying Computer Science and Computer Technology in the School of Communications & Informatics at the Footscray Park campus.

In learning any programming language, it is essential that students be given the opportunity to practice their new programming skills in an environment where they can receive constructive and corrective feedback. The provision of this feedback by staff is given directly during time-tabled laboratory sessions and also by the marking of regularly submitted exercises. The regular and routine assessment of these programming exercises is both mundane and very time-consuming. As Foxley et. al. point out, "marking is boring and assessment in general is probably the least liked task of most academics!" (Foxley et. al. 2001). Yet despite the heavy marking workload for those involved in the first year programming subjects, the value of regular and corrective feedback has been recognised as crucial to student learning of a programming language.

From a student perspective, feedback has not always been easy to get. When surveyed, students uniformly stated that the laboratory sessions were never long enough and that they were regularly competing with other students for a demonstrator's attention. In addition, our students and staff, like those of Arnow (Arnow 1995), have difficulty in synchronising schedules in order to supply help when the students felt they needed it.

Given that students would like more assistance and that staff goodwill was already stretched with heavy marking loads, one solution would have been to employ more staff and give students longer laboratory sessions and thus more feedback. In today's university environment the funding of this solution is very unlikely. Thus, against this background, the *submit* program was commissioned. Despite the existence of automated feedback and submission systems, we could not find one that was freely available, including source code, to enable full configuration and augmentation. We also required a greater degree of client platform independence than, for example, the BOSS system developed by Joy and Luck (Luck & Joy 1999).

The *submit* program has been developed so that students can receive instant and automated feedback to individual programming exercises. A student simply uploads their Java programs via a web page interface and receives immediate and automated feedback on various aspects including correct running on sample data, and on program style. Students are able to upload their submissions as many times as they like until the due date that has been set by the lecturer. After the due date passes, a tutor is able to review each student's final submission and provide further feedback, including a grade and comments on-line.

Section 2 of this paper contains relevant contextual information and some comparisons with programs similar to submit. Section 3 includes some technical details about the submit program. In section 4, the research method is described and Section 5 provides results of the trials we have conducted. Conclusions and further work are found in Sections 6 and 7.

2 Context

Footscray Park Campus of Victoria University is located in an ethnically mixed, traditionally working class area of Melbourne. It was the former Footscray Institute of Technology, before the amalgamation with Western Institute in 1992 which created the university. In 1998 the University was further expanded, by a merger with Western Metropolitan Institute of TAFE. Today there are 15 campuses and over 20,000 students enrolled in Higher Education and TAFE courses. The School of Communications and Informatics at the Footscray Park campus is responsible for the development and delivery of six Bachelor of Science, including Computer Science and Computer Technology and three Bachelor of Engineering courses, including Computer Engineering and Electrical Engineering.

Java is taught as a first programming subject to Computer Science and Computer Technology students. Currently, the Java programming teaching style is fairly traditional, with lectures, tutorials and laboratory work being the most common mode of delivery. Many students find difficulty in taking responsibility for their own learning and they have a high dependency on staff, not least because all laboratory exercises are corrected and assessed by hand.

In the laboratory, a tutor is expected to assist students in their programming endeavours and it can be a very hectic hour answering the rather repetitive and often trivial queries. There is never enough time to see every student during the hour and they are easily distracted if they are waiting for the tutor to help them.

Student feedback confirmed that most felt they needed longer time in supervised laboratory sessions. Most students commented that the style of problems were graded and covered the course adequately and these were not the issue.

On-line submission of student's work is totally new for staff and students in computing subjects at Victoria University of Technology. However, some universities such as the University of Melbourne have been using electronic submission of student programming projects for a number of years (Stern & Venables, 2002).

The submit program at the University of Melbourne is used essentially as an electronic mailbox for large, once or twice a semester project submissions. Students are able to receive electronic feedback against automatic testing, but they are often oblivious to invisible testing against which they are also marked. *submit* contains similar functionality to that of the University of Melbourne's submit program, but with a web interface, as opposed to running a shell script, and the additional option of specifying either text or graphics based programs. However, its direct incorporation into the teaching program to provide instant feedback to the students for their regular exercises is quite different in approach and execution from that of the University of Melbourne's program.

Another project developed at the Nottingham University was the Ceilidh System (Benford et al.,1994)(Zin & Foxley, 1994). It has been in use since 1988 marking student programs written in C. Although the programming language is different from *submit*, the reported aims of the Ceilidh project are similar. Yet the Ceilidh system achieves its on-line marking by restricting students to modifying and completing template programs. The more recent offering by Foxley et. al., the CourseMaster, (Foxley et. al. 2001) also places "strict constraints" on student's coding style. Thus student work conforms to a particular style thereby enabling the automatic testing and marking of their work. In contrast, our intention is the same as that of Joy and Luck, that is to allow students to submit assignments, and for the programs to be tested automatically (Joy and Luck 1998). It is not the intent of *submit* to do away with human feedback and perusal of student work.

In addition, there are a number of on-line teaching environments, such as WebCT (2002), which require staff to commit to remote and on-line teaching via a cyber classroom space. Again, the intent of using *submit* is to support students and staff in their normal face-to-face laboratory sessions, not to move interaction on-line. Also WebCT like programs offer much in the way of additional functionality ensuring they are complex to set up without offering the same Java-specific features that are required. Thus, after collecting the available options, a staff in-service day was held to evaluate several solutions, and *submit* was chosen as the most suitable for the situation.

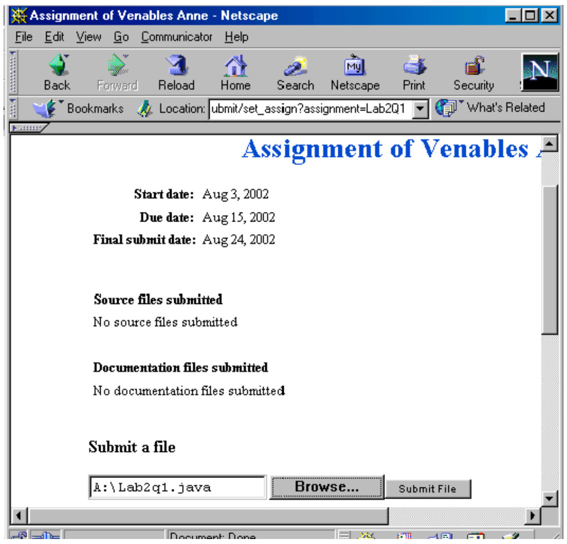
3 The Submit Program

submit is written in Java using servlets (Sun 2002), residing on a dedicated UNIX machine running a Java servlet platform, Tomcat (Apache 2002). The database backend is a hierarchy of flat files.

Java was chosen as the language of implementation as an experiment in itself for discovering the practicality of developing a serious Java application. It was also developed as an example for later year Java students to demonstrate techniques the students need to learn. The source code is freely available (VUT 2002).

The interface is intuitive, with minimal interaction and provides little option for mistaken input by students. Students find submitting their programs no more difficult than attaching a file to an email. Security, an issue highlighted by Reek (Reek 1989), and Luck and Joy (Luck & Joy 1999), is via login names and passwords already allocated to students by the School. Students must also be officially enrolled in a subject in order to login. A restricted execution environment, a Java sandbox, is used to execute the students programs to protect against uploading Trojans. This is an improvement over using the scheme described by Luck and Joy (Luck & Joy 1999), where the UNIX operating system is used to implement security which protects the file system, but not the network. In addition, groups of students can be formed via submit, eg. students working in pairs on an assignment.

Submission of a Java program is effected by entering the filenames (zipped files are automatically unpacked) of the class files. Clicking the "submit" button uploads the files to the server. Figure 1 shows the interface for this action.


Assignment of Venables Anne - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security
Bookmarks Location: ubmit/set_assign?assignment=Lab2Q1 What's Related
Document: Done

Assignment of Venables

Start date: Aug 3, 2002
Due date: Aug 15, 2002
Final submit date: Aug 24, 2002

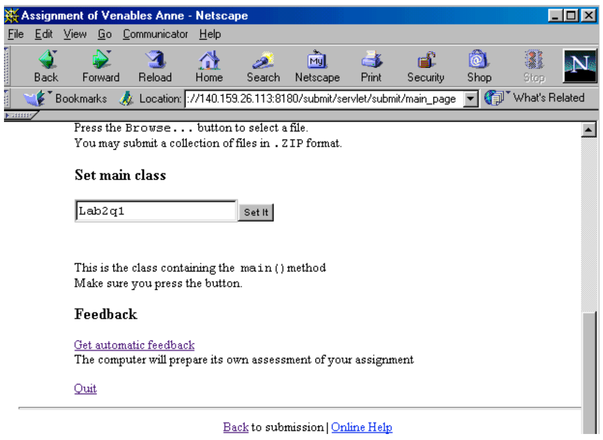
Source files submitted
No source files submitted

Documentation files submitted
No documentation files submitted

Submit a file
A:\Lab2q1.java Browse... Submit File

Figure 1: Submit a file

Once the files are submitted, the student must nominate a main class, as can be seen in Figure 2.


Assignment of Venables Anne - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: //140.159.26.113:8180/submit/servlet/submit/main_page What's Related

Press the Browse... button to select a file.
You may submit a collection of files in .ZIP format.

Set main class

Lab2q1 Set It

This is the class containing the main() method
Make sure you press the button.

Feedback

Get automatic feedback
The computer will prepare its own assessment of your assignment

Quit

Back to submission | Online Help

Figure 2 Getting automatic feedback

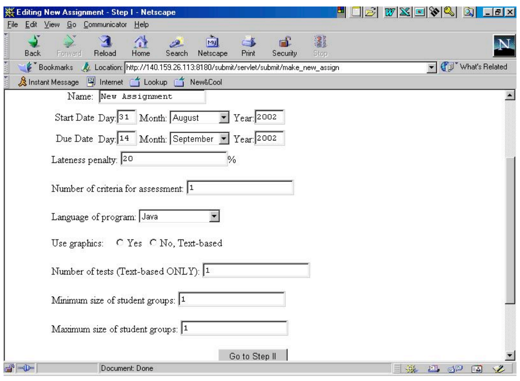
Clicking the "automatic feedback" link causes *submit* to do three actions. Firstly, *submit* analyses the student's program for elements of good style, such as comments and length of methods. Currently, this is confined to checking for comments at the beginning of each '.java' file and parsing the comment string to identify keywords: Name, StudentID, Date, TuteGroup, and Subject. A comment for each method is also expected, and a warning will be issued if a method is more than 50 lines long.

Step two is to compile the student's program on the server – compiler output is relayed back to the student. If compilation is successful, the student's program is run using the sample input provided by the lecturer. The program's output is displayed for the students to compare with model output as supplied by the lecturer.

Apart from the immediate feedback on compiling and running of a Java programming exercise, the students are also able to view tutor's comments and their mark on-line as soon as the tutor has graded it.

In addition to the student interface, there are interfaces for Tutors, to view files and grade assignments, and Lecturers, to set up new assignments, view various statistics, and review or re-assess student's submissions.

Setting up an assignment requires some input from the lecturer, such as due date, number and weighting of marking criteria, the type of output (text or as Java applet), and samples of input and desired output (if text). This is much simpler than writing scripts, as described by Reek (Reek 96), and the economies are the same, in that only one lecturer needs to set up the assignment and test cases.


Editing New Assignment - Step I - Netscape
File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Stop
Bookmarks Location: http://140.159.26.113:8180/submit/servlet/submit/make_new_assign What's Related
Instant Message Internet Lookup New&Cool
Document: Done

Name: New Assignment

Start Date Day: 31 Month: August Year: 2002

Due Date Day: 14 Month: September Year: 2002

Lateness penalty: 20 %

Number of criteria for assessment: 1

Language of program: Java

Use graphics: Yes No, Text-based

Number of tests (Text-based ONLY): 1

Minimum size of student groups: 1

Maximum size of student groups: 1

Go to Step II

Figure 3 Setting an assignment

There have been relatively few glitches in the *submit* program, and overall it has stood up very well during the trials described in Section 4. There are a number of enhancements that could be made to *submit*, as described in Section 7 and it is hoped that these will be effected in the near future.

4 Research Method

So far, there have been four trials of *submit*. The first trial was with a summer school cohort of students (2001/02). Students were required to submit their major programming assignment towards the end of the semester. There were 50+ enrolled students each submitting several different class files, using multiple attempts. This trial primarily tested that the useability of submit was sufficient, ie. that the interface was intuitive to new programmers. Performance of the hardware and software under normal usage was also under inspection.

The second trial, in first semester 2002 was designed to confirm that it was possible to use submit on a regular (fortnightly) basis for multiple laboratory exercises, as well as for major assignments. This trial involved a single lab group (out of 12 lab groups) to submit 20+ laboratory exercises (some consisting of several class files) during the semester. Unfortunately, the randomly chosen lab group was unusually small due to timetabling constraints. Each student in the cohort, including the group who used *submit* was surveyed about their laboratory sessions.

With a view to ultimately incorporating *submit* more fully across the curriculum, the third trial tested the use of *submit*, merely as an electronic letter box. This involved a group of 130+ Engineering students submitting only text documents that didn't require any of the "instant" feedback associated with submitting Java programs. The useability of the interface by students in courses other than Computer Science was also of interest.

A fourth trial, similar to the second trial, is currently under way with four lab groups, consisting of both new and repeating students. There are 70+ students each submitting regular Java programming laboratory exercises.

5 Results

The first trial demonstrated that students were able to successfully submit their Java programming assignment with minimal instruction, confirming that the interface is intuitive to use. In addition, the *submit* hardware and software handled the high demand usage near deadline time.

Principally, the second trial was to discover if the feedback supplied by submit to students was worthwhile and that they found the learning experience to be valuable. When students were surveyed, three responses came back fully completed. All three students were in agreement that the automatic electronic feedback was very useful although interestingly, all of these students stated that they had submitted only once. When asked about the ease of using *submit,* two students rated it as moderately easy and the third as very easy. When these students were asked how *submit* may be improved, one suggested a less complicated submitting procedure, another student stated that *'I don't see any other ways it should be improved'* and the third stated that *'The automatic electronic system is [a] very good way to submit'*

In addition, the second trial was concerned with evaluating the overhead of setting and grading the lab exercises on-line to see if it was too high for the lecturing staff. The setting of test cases for the automatic feedback did prove to be a measurable overhead (about half an hour per laboratory set of 5 or 6 exercises), especially as it was for only one small lab group. It was recognised that the same overhead would be incurred regardless of the size of the student cohort. Thus, if *submit* is adopted as planned for 2003, this overhead is acceptable for the 200+ students who will be using it.

The result of the second trial showed a reduction in the marking load traditionally associated in grading of exercises. Prior to *submit,* students handed in file printouts and the Java programs saved on disc. Considerable time was spent opening the disc, finding the directory and then the file, running it if student had remembered to compile it, if not, compiling the code then eventually running it several times against test cases, followed by reading the code in the printouts. Marking on line eliminates all the unnecessary handling of discs and locating and compiling of files, and running them against test cases. A tutor simply sees how a student's program has performed against the various test cases and at the click of a button is able to read the supporting code. The tutor is relieved of the above described drudgery and left solely to exercise academic judgement

The third trial was to see how *submit* stood up to the demands and possible abuse by a larger cohort (100+) of non-computer science students, submitting work that was not Java programs, but text documents instead. *submit* performed as well as expected, ie. without any drama, and the turn round time for grading work and providing feedback to students was reduced. Traditionally there had been a lot of paper shuffling and sorting of scripts into tutorial groups and placing the work in boxes ready for student collection with the added risk of mysterious disappearances of "good" work.

The collation of results has eased as *submit* provides a tab separated list of names, ids, and grades, ready for importing into a spreadsheet. This is a big improvement on the current system of each tutor creating their own list and the lecturer having to collate them at the end of semester.

The fourth trial is currently underway and will be finished at the end of semester 2, 2002. Indications so far, are that the benefits to the students are similar to that of the second trial, ie. Students are more willing to put the effort into improving their programs until *submit* is satisfied and that they obtain timely tutor feedback.

6 Conclusions

For students, the benefits of using *submit* include the fact that they are able to test their program by submitting it several times, hopefully, improving their program with each attempt. This has been evidenced by a program submitted on-line that included the following comment in the code.

```
/* Original Code
   goldCard.makePayment (100);
   System.out.println ("Balance owing: $" ….) ;

Modified code
Modified to conform with submit's expected
results
*/
```

By supplying output against various test cases student understanding of the requirements of the program is expected to improve before receiving feedback from tutors and demonstrators. This in turn has alleviated the laboratory tutors from the having to view student programs in order to provide the often repetitive feedback that *submit* can provide automatically on the simpler aspects of the student's programs. As a result, tutors have been able to spend more time helping students with less mundane questions and engage in discussion regarding style, strategy and algorithms.

Turnaround time for students retrieving tutor's comments and marks improved remarkably now that they are available on-line as soon as they are released. Previously, a turn around time of 10 days was typical, this has been reduced to 5 days.

The marking workload for laboratory staff has also been reduced. With *submit* handling the loading, compiling and running of all exercises, the staff need only assess academic judgement. A further reduction in workload for staff has also come about with the acceptance by students of automatic testing. Previously staff manually marked every exercise for every student. Now students happily accept that staff randomly chose one of the five or six submitted exercises for marking and the rest are taken care of by *submit.* This is a major time saver for staff.

There have been a few unexpected benefits. For some reason, students seem to be more willing to accept the deadlines chosen by the lecturer, but implemented by *submit*, and late submissions have reduced accordingly. Students also seem to be more willing to take responsibility for their learning, in that they will keep improving their program until *submit* accepts it without complaint, whereas previously, they had a tendency to be sloppy when it came to program style and testing. It has reduced the potential for confrontational discourse between tutor and student regarding programming style and testing.

There is also a role for *submit* in the policing of enrolments. Unenrolled students are able to attend class but can no longer submit their work for grading. There have been cases in the past where students have waited until they knew their grades before actually enrolling in a subject – thus saving on fees if they failed.

Our experiences have been similar to those of Joy and Luck (Luck & Joy 1999), however *submit* differs in the following important areas:

- *submit* is web-based. All levels of user (student, tutor, lecturer) have a web interface. There is no need to login to the host machine. The client machine needs only a browser and intranet connection.
- comparison with output is not made automatically, this is left up to the student to decide if their output matches the expected output.
- compilation is undertaken on the host machine, students cannot submit binaries.
- compilation is protected (as well as execution).

7 Future Work

There are a number of possible future extensions that can be made to submit, to increase both the robustness and functionality of the program. Currently, data is stored using a hierarchy of files. A better solution would be to store the data under a DataBase Management System, such as MySQL (2002). During the trials, the *submit* server has resided inside the University's firewall, since the computer systems manager was unsure of the security aspects if submit were to be open to the world. Ultimately, of course, it is desirable for students to be able to submit their programs from home.

Functionality could be increased by including an algorithm checker, such as proposed by MacNish (2000) and plagiarism detectors, such as those developed by Aiken (1994) and Joy and Luck (Joy & Luck 1999) .

Thought has been given to developing submit for other languages, such as C++, which is also taught in the school. One of the problems with C/C++ is the lack of Java security. One solution would be to have a dedicated machine to run student's programs that is separate from the server holding results.

Other possible improvements include:

- the incorporation of the assignment or lab exercise questions on the web page,
- an e-mail bridge to remind students who haven't submitted by the due date, or to e-mail results to students after the tutor has graded the work,
- integration with other campus systems such as automatically incorporating enrolment lists and submitting marks back to a central server.
- enabling tutor interaction with an application program, already possible for Java applets though this would require some client software, such as X windows.

8 Acknowledgements

We would like to thank Ian Haywood for his assistance in developing submit, Cameron Giles, and Foster Hayward for their part in procuring and maintaining a dedicated server; and Victoria University for supplying a teaching grant of $5,000 to enable the initial development of submit.

9 References

ACS (2002): Guidelines For Accreditation of Courses in Universities at the Professional Level, HTTP://www.acs.org.au.

Aiken, A. (1994): MOSS A System for Detecting Software Plagiarism, http://www.cs.berkeley.edu/~aiken/moss.html

Apache Software Foundation (2002): http://jakarta.apache.org/tomcat/

Arnow, D.M., (1995), :-) When you grade that: Using e-mail and the Network in Programming Courses, *In Seminars in Academic Computing Conference*, ACM.

Benford, S., Burke, E., Foxley, E., Gutteridge, N. and Zin, A. M. (1994):The Ceilidh System A General Overview. http://www.cs.cf.ac.uk/Dave/C/chapter2_19.html#appceilidh

Foxley, E., Higgins, C., Symeonidis, P and Tsintsifas, A. (2001): The CourseMaster Automated Assessment System - a next generation Ceilidh. *Computer Assisted Assessment,* University of Warwick, 5/6th April. http://www.ics.ltsn.ac.uk/pub/caa/index.html

Joy, M. and Luck, M., (1998):The BOSS System for On-line Submission and Assessment. http://www.ulst.ac.uk/cticomp/joy.html

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

MacNish, C. (2000) : Evolutionary programming Techniqies for Testing Students' Code. *Proc. of the Fourth Australasian Computing Education Conference,* Monash University, Melbourne*,* 170- 173.

MySQL AB (2002): Open Source Database, http://www.mysql.com/

Reek, K.A., (1989): The TRY System - or - How to Avoid Testing Student programs, In SIGCSE Bulletin 21(1), 112-116.

Reek, K.A., (1996): A Software Infrastructure to support Introductory Computer science Courses, In ACM SIGCSE Bulletin, 28(1), 125-129.

Stern, L. and Venables, A. (2002): Automated Tools to Support Small Group Teaching. Technical report. Department of Computer Science & Software Engineering, University of Melbourne.

Sun Microsystems Inc (2002): Java Servlet Technology, http://java.sun.com/products/servlet/

VUT submit home page: http://melba.vu.edu.au/~submit

WEBCT (2002): Leveraging Technology to Transform the Educational Experience, http://www.webct.com.

Zin, A. M. and Foxley, E. (1994) Automatic Program Assessment System. http://www.cs.cf.ac.uk/Dave/C/chapter2_19.html#appceilidh