CASE STUDY SEVEN
The BOSS System for On-line Submission and Assessment of computing assignments

Mike Joy and Michael Luck

The BOSS System for On-line Submission and Assessment of computing assignments

Author and contact:

Mike Joy & Michael Luck

Department of Computer Science

University of Warwick

Coventry, CV4 7AL

UK

Email:

M.S.Joy@dcs.warwick.ac.uk

Web URL:

http://www.dcs.warwick.ac.uk/Boss/

Key-words:

online submission, automated testing, programming

Outline:

Practical computing courses which involve significant amounts of programming continue to suffer from increasing student numbers. This makes their delivery and management more difficult to achieve effectively with the available resources. One solution to this problem is to develop methods for automating the submission and testing of student programs to support the marking effort and to enable the division of marking tasks among several individuals while ensuring consistency and rigour throughout. We have developed such methods in our system, called BOSS, and have successfully deployed different versions of it on several courses over a number of years. Here, we describe the original system and its recent enhancements, and discuss the benefits it has provided us with, both in terms of administration and in improving the learning process.

Context

The effects of ever-increasing student numbers continue to impact on the quality of course delivery in both further and higher education. This is especially true for computing courses in which there is a high practical programming component, and which are assessed by programming assignments. With large numbers of students, many lecturers are faced with the stark choice of investing vast amounts of time and effort in marking, through the separate processes of retrieving student programs, compiling them, running them and testing them manually against sets of test data, or of simply visually scanning a program and assigning a mark to it on the basis of a cursory assessment. The first of these options is becoming impossible due to the increasing demands on lecturers, yet the second is hardly acceptable.

By developing techniques for automating the submission, compilation and testing of student programs, we can support the process of marking, and enable marking tasks to be divided among several individuals while maintaining rigour and consistency. Not only can this stem the tide of an increasing workload, it can improve the learning experience available to students (in providing them with facilities for immediate and effective feedback, for example,) and also enable other administrative tasks to be automated as part of a coherent approach to full course management.

We can distinguish between courses which introduce specific programming languages, and those in which broader aspects of software engineering, such as techniques for design, methodologies, and analysis of algorithms are taught. It is not typically the purpose of introductory programming courses to cover such issues, which are often regarded as separate areas (Finkelstein, 1993). In considering the requirements for an automatic submission system, we focus specifically on just such introductory programming courses.

For these reasons, a programming assignment should be carefully and accurately specified. Without such a precise specification of the task, the assessment of programs can become significantly more difficult, as conformance to the specification may not be easy to judge. A tight specification, however, allows the submitted program to be tested against suitable test data, so that the output of the program can be compared with the expected output for each set of data. The requirement that solutions must conform to specification thus serves the dual purpose of enforcing good practice and enabling a measurable assessment of the program.

For several years, we have been using such a system that allows students to submit programming assignments on-line and to run those programs against test data (Luck and Joy, 1995). The system we have running is in its fifth year of use, albeit in different versions. It has been deployed on several courses, including those covering Pascal programming, UNIX Shell programming and C++ programming, each course attracting up to 200 students.

Description

The BOSS system for automatic submission of assignments (Joy and Luck, 1995) comprises a collection of programs, each of which performs a different task contributing to the overarching goal of effectively managing the process of submitting programming assignments on-line. BOSS is designed specifically for courses with large numbers of students, assessed by means of programming exercises.

The programs offer the following functionality:

q Students may submit programs on-line by means of a user-friendly program that conducts a dialogue with the student to ensure that the correct submission is made. The program is stored and simple checks are carried out so that the lecturer can subsequently test and mark it.

q All submissions for a specified item of coursework can be run against a number of sets of data. The output from the students' programs is compared with the expected output for each set of data. Time and space limits are placed on the execution of a program so as to prevent a looping program from continuing unchecked, and other steps are taken to minimise the potential for a program to damage the system.

q Submissions and the results of the testing process can be inspected on-line by authorised staff. Anonymity is preserved by storing data by University ID number.

q Students can test their programs by running them against one data set on which they will eventually be tested, and under precisely the same conditions. Thus a student can check that their program will run correctly under the final testing environment. This ensures that the program will work as the student expects when being tested and marked. In addition, it provides students with confidence that their submitted work does pass some minimal requirement.

q Final marks are stored in a SQL database and correlated with information from the University database (names and courses versus ID numbers and course registration, for example) to produce final marksheets for examination secretaries.

The BOSS system is a tool to allow students to submit assignments, and for those programs to be tested automatically. It is not an automated marking system. It is the responsibility of the individual lecturer to provide a marking scheme which takes account of the results produced by BOSS, together with all other factors which may be regarded as important (such as program style, commenting, etc).

Resource Implications

The BOSS system has provided us with a number of benefits without compromising the general approach taken of maximising exposure to standard tools and utilities. Large numbers of students have been handled efficiently by the system, with security of assignment submission being assured. Programs submitted cannot be copied by other students, and the possibility of paper submissions being accidentally lost is removed. Secretarial staff do not need to be employed at deadlines to collect assignments, making more efficient use of secretarial time, and the volume of paperwork involved can be reduced to (almost) zero both for the lecturer and for administrative and secretarial staff.

More importantly, perhaps, the time needed to mark an assignment is reduced considerably, while the accuracy of marking, and consequently the confidence enjoyed by the students in the marking process, is improved. In addition, consistency is improved, especially if more than one person is involved in the marking process.

Student Assessment

In this last year, we have also introduced the system into a second-year course which covers the practical application of software tools. Though this normally requires a slightly more involved testing regime, the BOSS toolkit provides a very adequate and appropriate means of automatic submission and testing. More importantly, perhaps, students have had virtually no difficulty in using the system. This seems to imply that the newly established culture has taken root, and that our initial efforts at integrating the system into the fabric of the degree courses are paying off.

Evaluation

As it stands, the system is functioning well. There has been a generally favourable student response, and this has improved as the culture of automatic submission has become established within the Department. In addition, lecturers and tutors have also found the system to be simple and easy to use, and marking times have been reduced significantly with a corresponding increase in consistency throughout.

We sought feedback from students by means of questionnaires which required students to comment on their experiences of using the system, and also questionnaires which required numerical responses for questions relating to system use. These were generally favourable, and most students considered it an easy system to use. The ability to use the utility to test programs in advance of submission to check the conformance of their programs to the specification was also widely appreciated.

The principal concerns expressed fell into two categories. The first of these covered minor criticisms about the user-interface and the specific messages that the system provides to students when a program fails the test utility. Many of these criticisms have since been addressed in the latest version of the BOSS system, and we are continuing development so that the user-interface is improved still further.

The second – and perhaps more interesting – category of criticism was that the output expected was too precisely specified. BOSS is far too "fussy". This criticism relates to the format of the output specified – as in the precise layout of tabular output, for example – and also to some students' desire to design their own user-interfaces by establishing interactive prompting for input, for example. This is an important point, for it seemed to reflect the preference of first year undergraduates who had had considerable programming experience prior to joining our course. Many of them were thus used to programming in an unstructured fashion and were unused to being required to follow precise specifications.

Key Advice

q Don't underestimate the value of a model solution to the construction of an unambiguous and precise problem specification.

q Use assessment tools for formative assessment when rapid feedback is required.

References

Finkelstein, A. (1993) 'European computing curricula: a guide and comparative analysis', The Computer Journal, 36(4):299–319.

Joy, M. and Luck, M. (1995) 'On-line submission and testing of programming assignments', in J. Hart, editor, Innovations in Computing Teaching, SEDA, London.

Joy, M. and Luck, M. (1996) 'Software standards in undergraduate computing courses', Journal of Computer Assisted Learning, 12:103–113.

Luck, M. and Joy, M. (1995) 'Automatic submission in an evolutionary approach to computer science teaching', Computers and Education, 25(3):105–111.