Survey feedback relating to the paper: Analysis of an automatic grading system within first year Computer Science programming modules

Emlyn Hegarty Kelly and Aidan Mooney
Department of Computer Science, Maynooth University, Maynooth, Co. Kildare. Ireland

emlyn.hegartykelly@mu.ie, aidan.mooney@mu.ie

Students, demonstrators and lecturers were asked for their feedback in relation to the automated grading systems used within the Department of Computer Science at Maynooth University in first year undergraduate modules. This document contains this feedback.

Feedback

1. Student feedback

The following is a sample of the feedback received to the question relating to how a student felt about using an automated grading system in the class:

- Getting feedback from prepared test cases instantly made experimenting with java easier.
- It's great and instant feedback is extremely helpful understanding the code.
- Very helpful and a useful tool.
- I think it's very useful for labs and personal use to practice.
- I feel it an excellent piece of software to practice my code.
- Very good and easy to use.
- Easier to plan and organise my code.
- I thought it was an innovative way to grade labs and assignments.
- Very beneficial and efficient.
- Useful in the sense that it tells you what your code is missing.

- Challenging, efficient, helpful to the module.
- Useful and quick for getting feedback/grades on lab questions.
- It was useful to see our scores in real time.
- Very useful. Having a built-in set of test cases meant the evaluation was a big help.
- It was very useful to see the proposed grade as it allowed you to make sure you were not missing anything.
- Helpful and motivates you to do the labs.
- It provides good feedback fast after evaluating the code.
- Most useful tool in the course.
- I like evaluating, a useful tool in ascertaining how close my answer was.
- Very useful to be graded on the spot and given a chance to improve.
- It is a useful method for quick assessment.

This small collection of comments presents a positive experience of the automated grading system experienced with the class. There was some negative feedback to the system which could be generally classified by the following feedback:

- It is difficult at times as you could have the code working but be marked down for small things, e.g. forgot a capital letter in what is printed out.
- It is good but would help more if it gave a more detailed description of errors.
- Worked well when not overloaded with users.

The first two negative comment were dealt with by improving the test cases and bash scripting to account for all possible scenarios. The final comment was an issue of resource management and was easily rectified. The feedback from students is in general very positive and encourages us to continue using automated grading in our lab sessions as it appears to have a positive impact on the students learning.

2. Demonstrator feedback

Demonstrators are present in each lab session, which lasts for three hours, and are assigned to work with up to twelve students.

The demonstrators were asked a number of questions and this section presents their feedback in relation to automatic grading systems used within the first year modules.

Q1: "From a demonstrator point of view what benefits do you see in using an automatic grading system"

- Because of the test cases it enabled myself and the student to see how close they were getting to the right output and what they were printing wrong.

- It's unbiased and gives a more accurate grade to the student.
- Allows for fairer grading, as the system is impartial. It also saves time where demonstrators can spend more time assisting students by answering their questions instead of grading them.
- Less biased and error prone correcting work, and automatic grading can resolve some trivial issues reducing demonstrator load.
- It gives instant and relevant feedback so students don't have to wait for the demonstrator to see if their work is correct/acceptable. It also frees up demonstrators so their time is spent more on helping students than marking their work.
- There's no bias towards any students since the grading isn't up to the demonstrator. Related to the above, students are less likely to ask the demonstrator to just give them full marks even though they haven't done the full work. There's less stress at the end of a lab because there's no rush to go around and make sure everyone is graded before the lab is over. Grading scripts are more likely to catch bugs than I would if I was just looking through/manually testing their code. It gives students a gentle introduction to IDEs. It can be stressful trying to set up and IDE when you don't know what you're doing. MULE allowed students to focus more on the code and then they can worry about tools in their own time.
- With auto grading, more time is allowed for teaching rather than marking their answers

Q2: "From a demonstrator point of view, does the feedback provided by the automatic grading system help you to help the students?"

- It helps me for the most part yeah! Quicker to see where they're going wrong.
- Yes, the system help me find the type of error quickly and the test cases help me determine what inputs are causing the students program to fail.
- Definitely.
- Yes, if the mistake was anticipated the feedback is good enough I don't need to help much, the students can make progress themselves. However, if the mistake was unexpected, it can be time consuming to find out where the problem lies .
- Yes, for compiler errors, the line and type are given in the terminal. For grading, if the student has left out a piece of essential code, they will be prompted on evaluation.
- Yes, failed test cases are very helpful to show the student where their code went wrong. Often the grading script will have better test cases than the student or I could have come up with it.

- Yes, with the feedback, reading errors is encouraged and this shows them how to locate their mistake, showing them how to read their errors has been helped greatly by the feedback

Q3: "From a demonstrator point of view, how useful do you find automatic grading in labs? (1- not useful at all and 5 - extremely useful)"

Figure 1 presents the feedback in a graphical form to this question. It can be seen the over 85% of the demonstrators found the

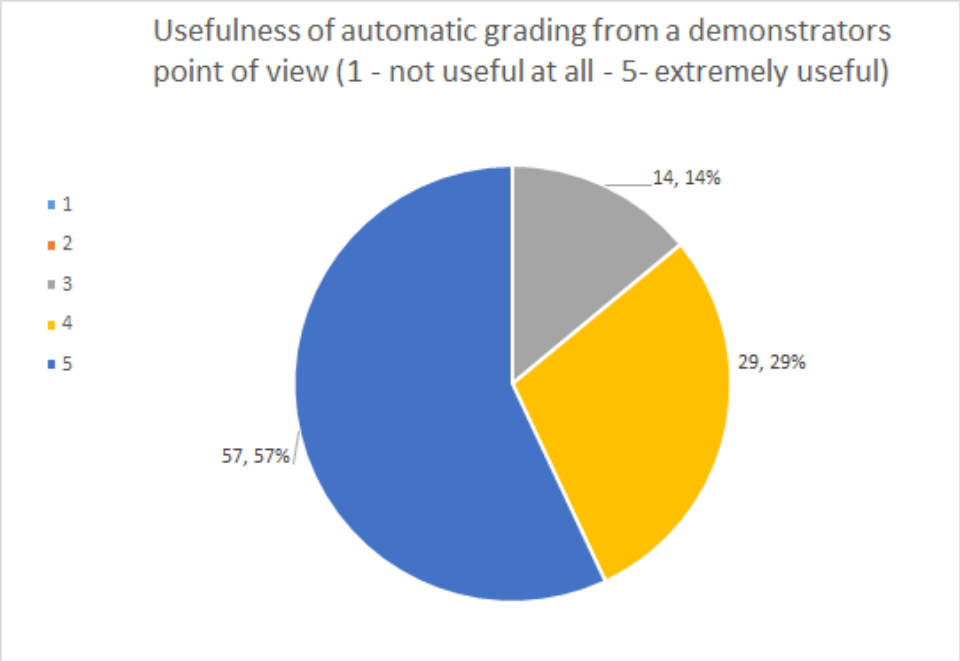

Figure 1. Perceived usefulness of automatic grading from a demonstrator point of view.

Some follow-on feedback relating to this question was also provided, which included the following statements:

- It allows the demonstrators to spend more time with the people that need help getting to the answers instead of spending most of the lab grading people who are already capable.
- It saves a huge amount of time by answering simple questions like "does this code work", and more complex test cases don't need to be typed out by each and every student.

- Even though there were problems during the semester I think the advantaged outweighed the disadvantages. The fact that there isn't a rush at the end of the lab to correct all the students is a big benefit because it gives me more time to help struggling students at the end of the lab. Test cases from the evaluation scripts often helped students to see the problem with their code without having to ask me for help. It's a nice introduction to IDEs and allows students to focus on the code more than the tool.
- On a whole the system is very useful, my only problem is the scripts it runs off requires very specific answers.

Q4: From a demonstrator point of view, what disadvantages do you see in using an automatic grading system?

- It can be unnecessarily picky and cause confusion e.g if the question wants you to print "Hello World" a student may get 0% for writing "hello world" but they've demonstrated they can solve the question.
- It can be very exact i.e the student may have working code but because they haven't an output that exactly matches the system, they often lose the majority of marks. There is only ever one correct answer and if it is not followed the student is harshly punished.
- The system is not always capable of guiding the student to the correct answer. A demonstrator grading can usually give an excuse for an answer being incorrect. There are cases where the system provides a fail, but the reason why is not obvious to the student, sometimes leading to further confusion.
- If there's a bug in the grading system, the whole lab is put on hold. If grading scripts have an error in them students can spend a lot of time trying to figure out what's wrong with their code even though it has nothing to do with them. Since I'm less involved with the grade that they get, I found that some students are less likely to ask for help and they end up spending more time stuck on one question. A lot of the help that I gave students this semester came from me asking them if they have any problems rather than the other way around.

3. Lecturer feedback

We also surveyed some lecturers who have used automated grading systems within their modules. Note, that some of the responses are relating to the use of automated grading tools on modules not at a first-year level.

Q1: "From a lecturer point of view, what benefits do you see in using an automatic grading system within a module?"

- "It allows demonstrators to spend more time helping students with their issues. It allows for students to be graded for the work they have done within their programs and not just on a right/wrong basis.It allows me to specify particular patterns I want in my code, for example, I can specify that there should be a for loop in the code and I can check that the pattern used by the student for a for loop is correct."
- It has the potential to provided faster feedback (sometimes dynamic feedback) to the student. It guarantees that grading is consistent. The digital recording of the of the submission and grade data is useful for auditing and storing of assignments. Once developed it can save time which can be used for demonstrators and lecturers to use in the lab for discussion.
- "Increasing the number of exercises, problems, and assignments which are completed by students certainly helps with student learning (In my own opinion). Increased number of assignments, however, coupled with increasing numbers sees an obvious increase in the amount of student work to assess (without doing more assessments). Feedback on these assessments is beneficial to learning, but it is becoming increasingly difficult to provide adequate, timely feedback to students in an non-automated way. While I have no empirical evidence of this I feel that there is also an increased expectation amongst the student cohorot for instant results/assessment/grades/etc. This is almost an impossible expectation to meet as students are not willing to actually consider the tradeoffs which are required. For the lecturer to deliver automated/instant grading there must be an understanding of the tradeoffs of this method from the student perspective namely that: (1) the test or assessment is structured in a specific way to allow for automated grading, (2) there are more, albeit smaller, assessments, (3) it can be very difficult to assess the attempt, the reasoning or the methodology provided in an answer. There might be a

simple but harsh right or wrong answer. In order to try and maintain some type of balance between teaching-assessment-research I feel that it is necessary to try to use automated grading systems where possible. Indeed, in the last few years I feel that there is greater workload all of the time in the design of assessments, correcting etc. This has the negative impact that the lecturer becomes a full-time teacher for almost all of the teaching parts of the semesters with little time left for research.

- Saves time and resources with respect to cost and also eliminates any bias. Timely - students receive grades and automated feedback immediately. Eliminates teaching and learning overhead delays from setup with respect to IDEs and environmental issues - different machines/laptops - different IDEs, different Java versions.

Q2: From a lecturer point of view, what disadvantages do you see in using an automatic grading system within a module?

- There is a time overhead in terms of creating the scripts.
- The approach can be inflexible, unable to react to unexpected but valid inputs. Unless a considerable effort is made in the design the approach focuses on test cases rather than the totality of the submission. To generate feedback other than a score is a challenge. Care needs to be taken that low scores generated by a binary scoring system are not demotivational. There is an administration overhead in fixing these issues such as correct answers that are slightly out of format are not being graded incorrectly.
- There is a balance to be found between the time taken to actually create the assignments, create the quizzes in Moodle, check that the quiz works, etc VRS the time taken to just create an assignment and then either grade it yourself or have the demonstrator team grade it. In the second case there is the additional overhead of creating suggestion solutions, marking schemes, etc. In terms of Moodle Quizzes (non MCQ type): There is no opportunity for the student's methodology or reasoning in an answer to be examined. The quiz just expects an answer x for a given question with a possible tolerance of d. So unless the answer is within [x-d,x+d] then the answer is wrong. The decision is a binary one. Depending on the question this could be harsh on the student as there might be a small miscalculation, logical error, etc which means that the methodology or concept is captured in their working towards the solution, but the end solution is incorrect. As some of the students mentioned to me this year ""the Moodle Quiz is brutally fair"". The fairness comes at the price of being brutal in the fact that all answers end up being either right or wrong with nothing for methodology. In SQL I have tried to address this by considering

questions which allow me to examine if the student really understands a particular concept and then can narrow down the search space for the answer down to something visible. For example, if we had a table with 1000 rows. The SQL question might be to find the DOB of the person with the shortest surname. The student can attack this question in several ways - the correct way MUST include somehow sorting the table and then ordering it by surname. WIthout this the answer would be impossible to find. Then they are asked to input the DOB exactly as it appears in the table (as a string) into Moodle. This year with Moodle (and to a lesser extent the internal database server) we have had many problems - connections timing out, losing network connectivity, etc. Student sympathy for these issues are ZERO as if these issues are a complete impossibility. On the side of the lecturer when these happen in a lab exam the clock continues to tick. With the lab timetable so densely packed and very heavy interconnectivity between modules and timetable slots there is often ZERO chance of 'doing the exam another time'. These problems mentioned here then mean that there is a need for manual rechecking of answers, dealing with a stream of student complaints and grievances, etc. The automated grading then becomes an additional burden on time resources. With a paper-based assessed which is corrected by hand there is none of these technical issues. Provided the lights do not go out and the heating stays on then the exam should be preseved. The many hours of preparing the automated grading system (checking answers, testing, etc) can be employed afterwards to do the correction. In summary:

  - Let A be the time required to setup, create and implement the automated grading assessment for a given class.
  - Let C be the time required to manually create and subsequently correct a manually graded assessment for a given class.

  The whole idea for me is that A << C. The disadvantages of automated grading outlined above mean that we begin to get a situation where A ~ C or in a worst case scenario A > C.

- There is the possibility that the REGEX pattern matching may overlook a correct or partially correct answer and a student may be marked unfairly however a manual check can resolve this quickly. The lack of teaching presence may be an issue where students may receive limited automated feedback or tailored feedback base on human intervention.

Q3: From a lecturer point of view, what improvements could be made to automatic grading systems?

- An easier method of creating scripts.
- It could be made simpler to construct an assignment, tools to do this are required. Feedback to the student could be more descriptive (comments as well as scores) and dynamic (as student does the work). Generate scores with finer granularity, rather than the binary correct/incorrect response. The system should look at the totality of the submission, code submitted, compile and run time and test cases. With code it should be aware of industry standards (e.g. indentation, commenting etc) and algorithms (e.g. detect quick sort or bubble sort).
- Again, in relation to Moodle. Allow for a wider range of text/string answers (which are obviously checked by more involved or complex regular expressions). Allow for the automated UPLOAD of questions and answers. It takes a bit of time to create a Moodle quiz. For example, N MCQ questions with 5 answers per question takes a good deal of time. It would be really useful and a timesaver you could upload these questions in JSON or XML or some structured text format. It would be far easier and quicker to create quizzes. I do realise that this isn't something that the majority of lecturers using Moodle would want.

Q4: "From a lecturer point of view, does the feedback provided by the automatic grading system help you to help the students?"

- Indirectly as it is allowing the demonstrators to provide help to them within spending time correcting students work.
- Yes, the quick feedback and time saved allows for more discussion in the lab sessions.
- I don't think so. It could be rectified with a simple common compiler error explanation sheet that students can consult.

Q5: "From a lecturer point of view, do you feel that the feedback provided by the automatic grading system helps demonstrators better assist the students in labs?"

- Yes definitely. The demonstrator can use the grading received by a student to direct their support and not get lost in trying to find errors within the code.

- VPL removes the need for the command line interface required by MASM (and C), this allows students to focus on concepts and for demonstrators to do the same. The quicker code/compile/review cycle also maintains focus on the assignment in the lab.
- Again - from a Moodle Quiz type of situation. The feedback provided, if any, is provided by myself. Personally, I tend to engage with my demonstrators frequently before labs. To ensure a more uniform 'message' being carried by the demonstrators around the labs I always ensure to meet my demonstrators before the lab, provide sample solutions with explainations, meet at the beginning of labs, etc. In this way I try to disconnect the demonstrators from the assessment of labs or assignments leaving them with more freedom to assist in helping students. Because of the automated nature of the grading any disputes arising are then directed to me (if they are not resolved quickly by the demonstrators themselves). Usually, the demonstrators aren't really aware of how the moodle quiz works in the background. They might not be aware of the number of answers assigned to each question or if there is a tolerance for an answer.
- Im relatively new to the system but I am not sure that the feedback provided by the VPL systems is fine grained enough to give specific details on the types of errors students experience beyond syntactic/semantic errors

Q6: "From a lecturer point of view, how useful do you find automatic grading in labs? (1 - not useful at all. 5 - very useful)"

An average rating of 4.5/5 was given here.