COMMUNITY DIRECTIONS

Teaching and Assessing at Scale: The Use of Objective Rubrics and Structured Feedback

Simon Grey[1] and *Neil Gordon[1]

1. Department of Computer, Science and Technology, University of Hull, Hull, East Yorkshire, United Kingdon,

*Corresponding Author: n.a.gordon@hull.ac.uk

Keywords: *Computing Education; Feedback; Teaching Programming*

Abstract

It is widely recognised that feedback is an important part of learning: effective feedback should result in a meaningful change in student behaviour (Morris et al., 2021). However, individual feedback takes time to produce, and for large cohorts – typified by the North of 300 challenge in computing (CPHC, 2019), it can be difficult to do so in a timely manner. On occasion it seems that many academics lose sight of the purpose of feedback, and instead view it to justify a mark, rather than an opportunity to provide meaningful tuition. One strategy to provide feedback at scale is to share the workload across multiple staff, but this introduces an additional problem in ensuring that the feedback and marking are equitable and consistent. In this paper we present a case study from teaching programming that attempts to address two distinct, but related issues.

The first issue is to make feedback more meaningful. We attempt to achieve this by providing detailed feedback on a draft submission of programming coursework allowing students time to make changes to their work prior to the final submission date. We present an analysis of the data generated from this approach, and its potential impact on student behaviour.

The second issue is that of scalability. This feedforward approach creates a significant pressure on marking and on the necessity to provide feedback on a draft submission to large numbers of students in good time so that students can act upon it. To achieve this, we consider an approach based on creating an objective, reusable marking rubric so that the work can be reasonably spread across multiple members of staff. We present an analysis of the data generated from this approach to determine whether we consider the rubric to be objective enough to remove individual interpretations and biases, and where discrepancies exist attempt to determine where those discrepancies arise.

This work was carried out through an analysis of impact on student assessment, as well as from the academic staff involved in using the rubrics. Preliminary results from this work show that the more objective rubric used by several did enable a scalable solution for rapid feedback on submissions, and this did indicate some improvement in student outcomes. However, the work also illustrated the problems of subjective interpretations and some variation in outcomes by marker.

Introduction

With an increase in the number of students choosing to study in many disciplines, and in particular computer science, there is an ongoing challenge in how to effectively teach and assess in a scalable way. Within computer science and many other sciences, the demand for introductory courses in programming is increasing. As with other competencies within science, programming is a difficult skill

https://doi.org/10.29311/ndtns.vi19.4103

to learn which requires a significant amount of dedicated and deliberate practice to master. Providing appropriate and timely feedback can be invaluable in accelerating learning (Morris et al., 2021).

The teaching of programming is one of the core features of computer science degree programmes, indeed it is one of the key competencies for computing education. However, the teaching of programming is an acknowledged challenge, with computer science typically demonstrating some of the worst outcomes of any discipline (Gordon, 2016).

Feedback is a key tool in learning and teaching. It is important to students and to institutions, as illustrated by its prominence in the National Student Survey, where the focus is on students' perceptions of feedback (McKay et al., 2019).

This paper considers how to potentially improve student outcomes through a scalable approach to feedback for a programming module using reusable, objective rubrics for an introductory programming module. Whilst based in the context of computing, several aspects are transferrable to other topics and disciplines.

The paper explains the rationale of the approach for feedback in programming, how to share feedback across a team, and its impact on student's behaviour and outcomes. The paper analyses some of the data generated from feedback provided for students work from multiple markers, and we consider whether the attempt to create an objective rubric was successful.

The Learning Cycle-Why

Feedback is an important aspect of learning and teaching. Feedback is the primary mechanism by which students can be provided with guidance on how far their work is meeting the expectations of their teachers, and in that way can aid student's self-reflection. Kolb's experiential learning cycle (Buress & Hanshaw, 2006) included reflection as part of the way that students can learn. This is especially relevant in the computing discipline. Kolb's cycle can be adapted for computing and mapped to a typical programming style practical learning activity where students explore their understanding of the concepts by producing software and then getting feedback on how correct they seem to be. **Figure 1** shows an example of this type of approach to an iterative learning process. The challenge in large classes is how to properly close the feedback loop, which implies that effective feedback results in a meaningful change in student behaviour.

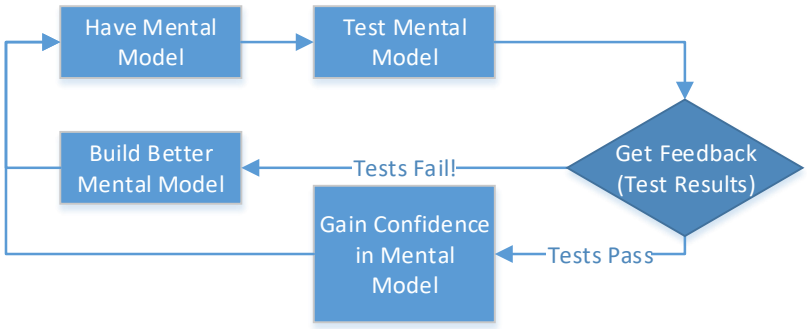

Figure 1 Developing a mental model of the work

Feedback-Previous Work

As noted already, there is a significant emphasis on feedback in higher education. However, it is worthwhile to take a moment to consider what feedback is for.

https://doi.org/10.29311/ndtns.vi19.4103

With increasing demands on staff and with feedback often being delivered sometime after learning has notionally finished, feedback could be seen as a justification to help a student make sense of the mark they have been awarded. However, in a meta-analysis of different types of feedback, Hattie and Timperley (Hattie & Timperley, 2007) discovered variability in the effectiveness "those studies showing the highest effect sizes involved students receiving information feedback about a task and how to do it more effectively." They proposed a feed-up, feedback, feed-forward model, highlight "the importance of ensuring that feedback is targeted at students at the appropriate level".

Where does learning happen?

Learning of difficult concepts and techniques is cultivated through deliberate practice (Ericsson et al., 1993). This is more effective where the personal challenge is closely aligned to the learner's skill level and that they have a growth mind-set (Dweck, 2016), which perceives failure as an opportunity. This can be effective in computing education (Pembridge & Rodgers, 2018). Considering where the learning happens, it is only by rigorously testing our mental model, finding where our mental model fails, and adjusting accordingly that we learn. This is something that we also see in the practices, processes and tools used in industry. By developing better techniques to test reliably and iterate faster, industry is able to perform more iterations on a problem and develop better solutions.

In this approach to learning, students initially build a mental model of whatever topic they are considering. They test the model and observe the feedback from that test. If their test passes, they build confidence in their model, which is good. If their test fails, that is great! They have found a flaw in their mental model and then revisit the original problem as they develop their mental model.

This paper outlines how this was used in a first year (level 4) introductory programming module, for approximately 200 students. When preparing the module, it is assumed that all students will engage with the module; however, typically a significant number of students will not engage for various reasons. For example, they may never turn up at the University, transfer to different course or withdraw from their studies altogether.

Rubrics and assessment

Using a rubric has the potential to provide students with a structured form of feedback (some prefer the phrase feedforward as more indicative of the intent), in a way that can enable them to improve their future work. Ahoniemi and Karavirta (2009) analyse the use of a marking rubric to achieve objective marking and quality feedback and provide an excellent description of the conflicting forces impacting the teaching of programming, as well as some ideal solutions. They highlight how the functionality of a program can often be tested using automatic tools, whilst the quality of code is harder to assess, and the need for rich, timely and specific feedback is required to help students improve code quality. They also discuss how the use of an objective rubric can be used to achieve parity between markers, and demonstrated how objectivity can be improved over time by using the same rubric over time. Becker (2003) describes the use of two detailed rubrics to help provide objective feedback at scale for students in first year programming classes. The use of two rubrics enables a general rubric to be used for many assignments, whilst a second rubric assesses functionality specific to a particular assignment. Becker notes that the use of a consistent general rubric will enable markers to become more familiar with the marking scheme over time, increasing their marking efficiency.

Feedforward Assessment

The objective of this research is to develop a solution that can be used to provide rich and meaningful feedback on a practical programming activity, one that can be provided in a timely manner to a large cohort on of first year computer science students. The intention was to develop a solution that is scalable, objective and reusable. This was intended to provide feedback that results in a change in student behaviour, meaning students need feedback with enough time to understand the feedback, make changes and get a better summative result (following the flow shown in **Figure 1**).

https://doi.org/10.29311/ndtns.vi19.4103

One of the key constraints is time, and the complexity of the assessment and analysis of it for marking can make it difficult to automate feedback. The solution adopted was to parallelise feedback generation across staff.

As identified above, the rubric design was intended to provide timely, scalable, objective feedback, with a reusable rubric. This built on some of the earlier work mentioned, with an abstract and functional rubric (Becker, 2003) that was objective and provided parity (Ahoniemi & Karavirta, 2009). This is described in the next section.

Timely and scalable

There is significant pressure to get feedback in time for it to be meaningful, during the teaching period e.g. within a 10- or 12-week semester. Large classes are a challenge in many disciplines. Many computing departments have been dealing with rapid and significant growth leading to the “North of 300” challenge (CPHC, 2019). The need to make scalable marking is key. This requires a marking scheme that can be applied by several markers in parallel, with relatively rapid decisions on criteria and marks.

Objective

A rubric needs to be objective, so that marking can reasonably be parallelised and offers parity to students. This means avoiding ambiguous terms like “Good” or “Very Good”. To be meaningful for feedback, it should be accessible to students so they can understand the content. This can be done by using terminology and descriptions that are clear, though preferably do not lead to unfounded appeals where students believe they have achieved a criterion that the marker has not recognised.

Reusable

To allow for manageable workload, and parity in a longitudinal sense between cohorts, an aim is for parity year on year. In the context of programming teaching, this means that sections on abstract functionality and code quality should persist year on year. Whilst it is appropriate to set new assignments for students each year to minimise the temptation for students to potentially engage in unfair means, it is also desirable to create a rubric that can be used in subsequent years. This not only reduces the amount of work required to create new assignments, but also ensures that assignments are comparable year on year and can allow staff to improve their marking efficiency.

To achieve this goal, assignments were created based upon a rubric with three distinct subsections. The first subsection is functional and is concerned with whether the solution fulfils the specific task. This section is worth 30% and would have to be replaced with each new assignment, as each new assignment necessitates new functionality.

The remaining 70% are based on abstract functionality and code quality. Abstract functionality is worth 15% and encompasses elements that can be included in every assignment, that address abstract concepts and is written in an abstract way where the implementation will depend on the assignment itself.

The remaining 55% of the assignment is concerned with code quality. This is important because it is possible to create a functional program in many ways. The code quality portion of the rubric encourages students to focus less on whether their code works and shift more towards whether their code is well written. An example of the rubric breakdown for a programming assignment to develop a word search is given below in **Table 1** and **Table 2**.

https://doi.org/10.29311/ndtns.vi19.4103

Set up board	5%
Display Wordsearch	10%
Enter Coordinates	5%
Give Appropriate Feedback	5%
Detect win, reset and play again	5%

Table 1 Specific functionality (30%)

Use of Console Based Menus	5%
Saving to and Loading from a File	10%
Use of Methods	15%
Use of Structured Data	15%
Exception Handling	10%
Self-Commenting Code	5%
Use of Explicit Comments	5%
Use of Source Control	5%

Table 2 Abstract functionality and code quality

Within each criteria, marks are awarded across a scale divided into between 5 and 8 grades, each accompanied by a detailed objective descriptions of what is required to be awarded the marks associated with that level using language appropriate to that section. For example, in the enter coordinates criteria, 2 points are awarded if the implementation “can enter rows and columns, but no unexpected entries (letter and values out of range) are handled”.

It is estimated that marking each student’s work would take between ten and twenty minutes. For the 214 students on the module, this would take between 36 and 71 hours. This is a significant undertaking, especially alongside other teaching happening during the semester. For this reason, the marking and feedback task was shared between four members of staff (See **Table 3** for the allocation of marking).

Marker Number	Students in Range	Drafts Submissions Marked	Final Submissions Marked
1	53 (A-C)	15	36
2	53 (C-K)	18	32
3	53 (L-R)	22	37
4	55 (R-Z)	17	34

Table 3 Marking allocation between markers

Students were given the opportunity to submit a draft by a specific date. After that date staff would mark the drafts and provide written feedback. A total of 80 students submitted draft work for marking. For pragmatic reasons concerning the marking interface staff, each member of staff was asked to mark a range.

Feedback on drafts was returned to students one week after the draft submission, and one week before the final submission. Students were also invited to communicate with the member of staff who gave their feedback to seek any clarification on the feedback they were given.

https://doi.org/10.29311/ndtns.vi19.4103

Does it work- Rubric Efficacy

If the rubric was objective, we would expect the marking profiles to be similar. The overall average of all marks was 56%. The average marks for individual markers is shown in **Table 4**.

Marker Number	Final Submissions Marked	Average Mark (%)
1	36	48
2	32	66
3	37	54
4	34	58

Table 4 Average Marks Awarded By Each Marker

In a single factor analysis of variance (ANOVA) test the P value is 0.134 which is greater than 0.05 so we accept the null hypothesis that there is no difference between the means of the 3 groups. However, the F value is 1.89 and F critical is 2.67 – so the result is not significant.

Figure 2 illustrates the profiles for the markers in terms of their marking range and averages.

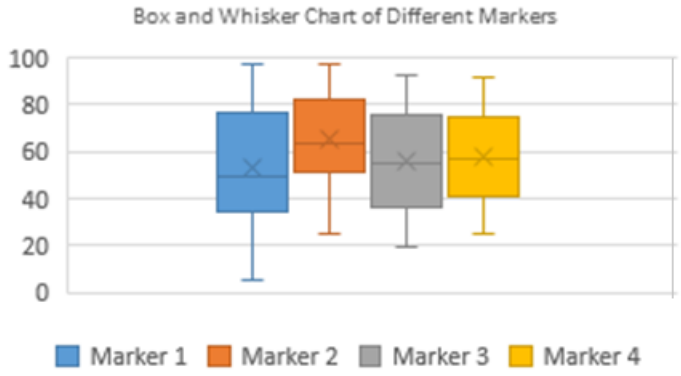

Figure 1 Box and whisker chart showing differences between markers

Figure 3 illustrates the marker profiles and shows that there was some discrepancy here. The use of a (more) objective rubric did support the idea of multiple markers – without redress to a separate moderation process after marking

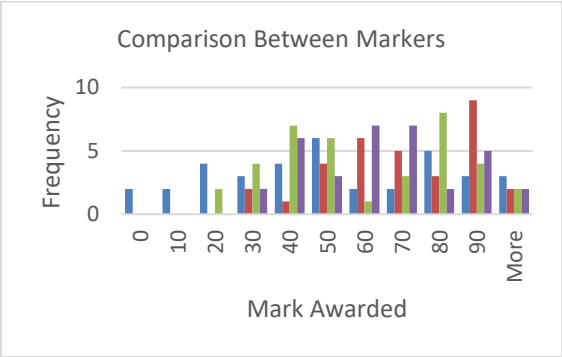

Figure 2 Rubric Efficacy – Marker Profiles

https://doi.org/10.29311/ndtns.vi19.4103

The profile of the marking shows similarities for 3 of the 4 markers. This indicates some success in objective marking. The disparity for the 4th (outlier) does show that subjective issues are hard to rule out.

Feedforward Efficacy

In terms of the impact on students, 80 students submitted a draft for feedback whilst, 59 students did not submit a draft. Considering the marking profile of first attempts and final attempts in **Figure 4** and **Figure 5**, we can attempt to understand the impact of the rubrics and rapid feedback.

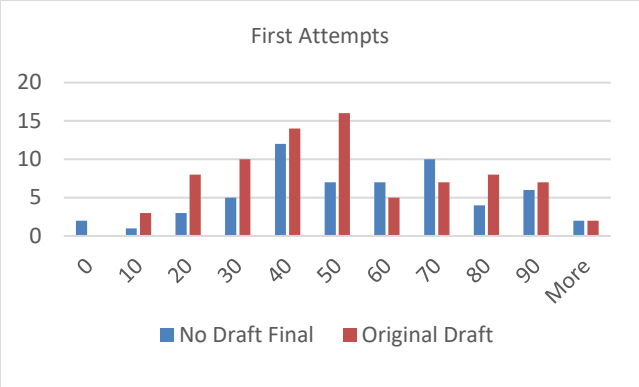

Figure 3 Feedforward Efficacy – First Attempts

Figure 4 shows the first attempts of all students. The blue (left hand column in each pair) is the first and only attempts for those students who did not submit a draft. The second (red, right hand) column in each pair is the first mark (first feedback) of those who took the opportunity to submit a draft. This indicates a similar profile, though there appears a slight skew to higher marks in in the draft versions of those who took advantage of the opportunity for feedback on a draft: this is most marked between the bare pass (40%) and 50% mark range.

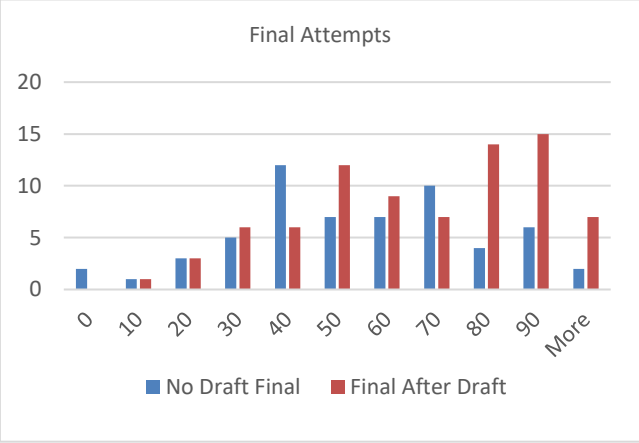

Figure 4 Feedforward Efficacy-Final Attempts

Figure 5 illustrates the difference in the final mark profiles between those who did not submit a draft (the blue, left hand column in each pair), and those who submitted a draft (the red, right hand in the pairs of columns). This shows a clear difference between the two groups and supports the ideas that those who submitted drafts did learn from the feedback and benefitted through higher final marks.

https://doi.org/10.29311/ndtns.vi19.4103

Conclusions

The objective and scalable rubric approach enabled timely feedback in a large module. The take up of the chance for feedback on a draft was fair though still disappointing. The profile in the previous draft does show that those who chose to submit a draft had a higher final mark profile, and their profile changed from the normal profile in the first attempt to a skewed (positive) profile in the finals. The rubric and feedback mechanism for the assessment shows the benefits where students do engage with feedback and should be transferrable to other contexts and institutions.

References

Ahoniemi, T. and Karavirta, V. (2009). Analyzing the use of a rubric-based grading tool, *ACM SIGCSE Bulletin*, 41, 3, 333-337.
https://doi.org/10.1145/1595496.1562977

Becker, K. (2003). Grading programming assignments using rubrics, *Proceedings of the 8th annual conference on Innovation and technology in computer science education,* 253-253.
https://doi.org/10.1145/961511.961613

Burgess, G.A. and Hanshaw, C. (2006). Application of learning styles and approaches in computing sciences classes, *Journal of Computing Sciences in Colleges*, 21, 3, 60-68.
https://dl.acm.org/doi/10.5555/1089182.1089190

CPHC (Council of Professors and Heads of Computing). (2019). North of 300: Dealing with Significant Growth.
https://cphc.ac.uk/2019/01/08/north-of-300-dealing-with-significant-growth/

Csikszentmihalyi, M. (2014). Toward a psychology of optimal experience, *In Flow and the foundations of positive psychology*, 209-226, Springer, Dordrecht.
https://doi.org/10.1007/978-94-017-9088-8_14

Dweck, C. (2016). What having a “growth mindset” actually means, *Harvard Business Review*, 13, pp.213-226.

Ericsson, K.A., Krampe, R.T. and Tesch-Römer, C. (1993). The role of deliberate practice in the acquisition of expert performance, *Psychological review*, 100, 3, 363.
doi: 10.1037/0033-295X.100.3.363

Gordon, N.A. (2016). Issues in retention and attainment in Computer Science. York: Higher Education Academy.

Hattie, J. and Timperley, H. (2007). The power of feedback, *Review of educational research*, 77, 1, 81-112.
https://doi.org/10.3102/003465430298487

MacKay, J.R., Hughes, K., Marzetti, H., Lent, N. and Rhind, S.M. (2019). Using National Student Survey (NSS) qualitative data and social identity theory to explore students’ experiences of assessment and feedback, *Higher Education Pedagogies*, 4, 1, 315-330.
https://doi.org/10.1080/23752696.2019.1601500

Morris, R., Perry, T. and Wardle, L. (2021). Formative assessment and feedback for learning in higher education: A systematic review, *Review of Education*, 9, 3, 3292.
https://doi.org/10.1002/rev3.3292

Pembridge, J.J. and Rodgers, K.J. (2018). Examining self-efficacy and growth mindset in an introductory computing course, *IEEE Frontiers in Education Conference (FIE),* 1-5. IEEE. https://doi.org/10.1109/FIE.2018.8658728