Measuring the impact of high quality instant feedback on learning

Practitioner Research
In Higher Education
Special Assessment Issue

Vol 10(1) pages 130-139

Stephen Nutbrown, Colin Higgins, Su Beesley*
University of Nottingham, Nottingham Trent University*
syn@cs.nott.ac.uk

Abstract
This paper examines the impact of a novel assessment technique that has been used to improve the feedback given to second year Computer Science students at the University of Nottingham. Criteria for effective, high quality feedback are discussed. An automated marking system (The Marker's Apprentice - TMA) produces instant feedback in synergy with the highlighted best practises. This paper investigates improvements in the work submitted by students after receiving this type of feedback. It draws upon surveys, as well as comparisons to previous cohorts for validating the positive impact of these techniques. It was found that the cohort (141 students) made 35% fewer common mistakes on a subsequent exercise than the previous cohort. 100% of students surveyed (35) claimed to have read the feedback, and 91% agreed it clearly identified areas for improvement. 68% agreed the feedback is useful for their learning, which is backed up by the improvements seen on the last exercise. Supported by these results, this paper concludes that following feedback best practises can have a substantial impact on learning, and that automated assessment can play a key role in providing this instant, high quality feedback. The results and similar techniques can be applied to other disciplines.

Keywords
Assessment; feedback; feedfowards; automated assessment; feedback engagement.

Introduction
Assessment is a cornerstone of learning in higher education, and it is claimed that feedback is 'more strongly and consistently related to achievement than any other teaching behaviour' (Bellon, Bellon and Blank, 1992). Studies show that innovations designed to strengthen the frequent feedback that students receive about their learning yield substantial learning gains (Black and William, 1998). However, NSS scores highlight student dissatisfaction with feedback, with the lowest scoring question on the entire survey for full time taught courses at registered higher education institutes being 'feedback on my work has helped me clarify things I did not understand' (HEFCE, 2015). The third lowest scoring question was 'Feedback on my work has been prompt'. Studies also suggest that many students fail to engage with the feedback that is provided to them (Duncan, 2007).
This study considers the feedback which is given to students for programming assignments and the impact it has on the quality of subsequent work. First, past research is reviewed to define a set of criteria that 'high quality' feedback should meet. Then, an introduction to 'The Marker's Apprentice' (TMA) describes the type of feedback that can be instantly generated for student submissions, and discusses how the feedback meets the criteria discussed. The results of 346 submissions from 60 unique students to a second year programming assignment (multiple submissions were allowed) allow improvements made by students to be studied. Considerable improvements between the first and last submissions are observed, and a survey explores students' views of the feedback. These improvements suggest some form of learning has taken place. However, this paper questions if students have been mechanically fixing the issues that TMA highlights, or if deeper learning has

taken place. To explore this, the impact on the subsequent assignments completed by the same cohort is analysed. The students are reassessed on the following exercise, this time without allowing multiple submissions. The results allow the comparison of the current year (2014/2015) to a historic cohort (2012/13) who completed the same assignment, but had not benefited from the type of feedback produced by TMA. The module convenor, lectures and coursework assignments between the two cohorts remained relatively constant. This direct comparison gives us insight into the impact of receiving instant high quality feedback. For the comparisons and surveys, ethical approval has been granted through the University of Nottingham's ethics review process.

Criteria for 'high quality' feedback

Academics have put considerable effort into defining methods and criteria for effective feedback; the focus of this paper is the application of these practises. There are a common set of properties and definitions of good feedback that repeatedly appear in surrounding literature. Feedback should be:

- Timely. If feedback is not timely, students may not take the time to go back and review the assessment which may seem distant and remote. Timely feedback promotes student engagement. Research has shown the sooner feedback is received, the more effective it is for learning (Irons, 2008).
- Informative. In a study by Higgins et al. (2002) 'what can be done to improve' and 'explaining the mistakes' were rated by students to be the most important types of feedback. Where possible, feedback should be clearly linked to learning outcomes and should give some form of action relating to how to improve (Beesley et al., 2014).
- Reliable and consistent. Feedback (both summative and formative) should be reliable and re-testable. If a submission is re-tested, it should receive the same grade and feedback. The assessment cannot be valid if the results are not reliable.
- Clearly communicated. Feedback is of no use if it cannot be understood. In a study by Handley et al. (2007), a quarter of the students surveyed had problems reading the module convenors handwriting. This is not a finding in isolation, similar suggestions were made by students surveyed by Beesley et al. (2014).
- Feedback should be specific. Studies have reported students often expressing uncertainty with vague or ambiguous feedback, e.g. 'It just says presentation. You don't know if it is our presentation, the way we were dressed, or something, it could have been anything'.(Price et al., 2010).
- Useful for teachers. Feedback and assessment is not just for students. Yorke (cited by Nicol and Macfarlane-Dick, 2006) notes 'The act of assessing has an effect on the assessor as well as the student. Assessors learn about the extent to which they [students] have developed expertise and can tailor their teaching accordingly'.

The Marker's Apprentice (TMA)

The marker's apprentice is an electronic assessment system built for automatic, semi-automatic and manual assessment of student submissions. It is a new system built upon the lessons learnt from the development and experience using CourseMarker (Higgins et al., 2005). It allows students to submit assignments electronically via a web browser. The marking criteria for an assignment are divided into parts, each of which has an associated tool. Tools may be manual (e.g a marker can type in a grade and some feedback), semi-automated (e.g tools allowing a marker to pick from a set of grade/feedback options), or fully automated. If the assessment consists only of automated tests, it is possible for the grade and feedback to be instantly presented to the student. This paper focuses on the instant feedback provided by one of the automated tools for the assessment of programming. For the assessment of programming there are two main areas for consideration. First, functional correctness is a measurement of whether the program does what it is intended to do. This can be measured by a series of unit tests or input/output tests. These tests execute the student's program

with some predefined input, and validate the output against what is expected. Secondly, a tool is used to measure the quality of the source code. It is possible to write code which functions correctly, but due to its structure can be exceedingly difficult to read. Poorly structured or difficult to read code leads to maintenance problems and increases the difficulty of identifying and fixing mistakes. The structure of source code is highly important and is often cited as an important issue for students graduating and moving into industry. Many books and literature has been written on the subject, 'even bad code can function, but if code isn't clean, it can bring a development organization to its knees' (Martin, 2008).

The Marker's Apprentice utilises a set of 127 rules for measuring bad practises which result in messy, difficult to maintain code. These are rules that are commonly used in industry to identify bad coding practises. The rule definitions and detection methods are from a large, well-supported open source project (PMD, 2015). These rules vary from naming conventions (e.g naming variables 'a', 'b', and 'c' for instance makes code much more difficult to read than if they are named 'min', 'max' and 'average') to measuring the correct use of braces on if-statements and the complexity of student solutions. These rules are grouped into rulesets relating to particular programming constructs that students are expected to use. For example, one ruleset groups all rules to do with loops, and another groups those to do with switch statements. With these rulesets and using PMD, TMA is able to identify if, and exactly where in the student submission individual rules have been violated. This testing is extremely fast (seconds), and results are given back to the students showing the details of each rule violation, where it occurred, a description of why it is a bad practise, an example of it done well and poorly, and finally a link to an external resource which may be lecture notes or academic references. Before TMA, these aspects were marked by hand and were susceptible to issues of consistency and long turnaround times.

Table 1. shows a pair of 'bad' and 'good' extracts of code which may appear in a student submission.

Table 1. A bad and good example of code which does the same task to highlight different coding styles.

Bad code, example with common issues	Good code, example without common issues

int a = 99;
int b = 10;
int c;
 	
if(a < b) c = b-a;
else if (a > b) c = a-b;
else c = 0;
	int first = 99;
	int second = 10;
	int difference;
	
	if (first < second) {
	  difference = second – first;
	} else if (first > second) {
	  difference = first - second;
	} else {
	  difference = 0;
	}

Note the good example is easier to read. Mistakes include variable naming and missing braces on if-statements.

The feedback returned to the student includes a bar chart is drawn to illustrate how they performed on each part of the assessment. For each part, a list of rule violations is shown with a short description and the location it was found in the code (figure 1). For each violation found, a link to a

short tutorial is provided (figure 2). The short tutorial gives details of a good example, a bad example, together with a description and (optionally) a link to learning resources or further reading. The bar chart provides an overview, to allow the student to easily identify areas of concern, whilst the list of violations and short tutorials provide information specific to each issue. Students can interactively click on each bar in the bar chart to view any issues related to that part.

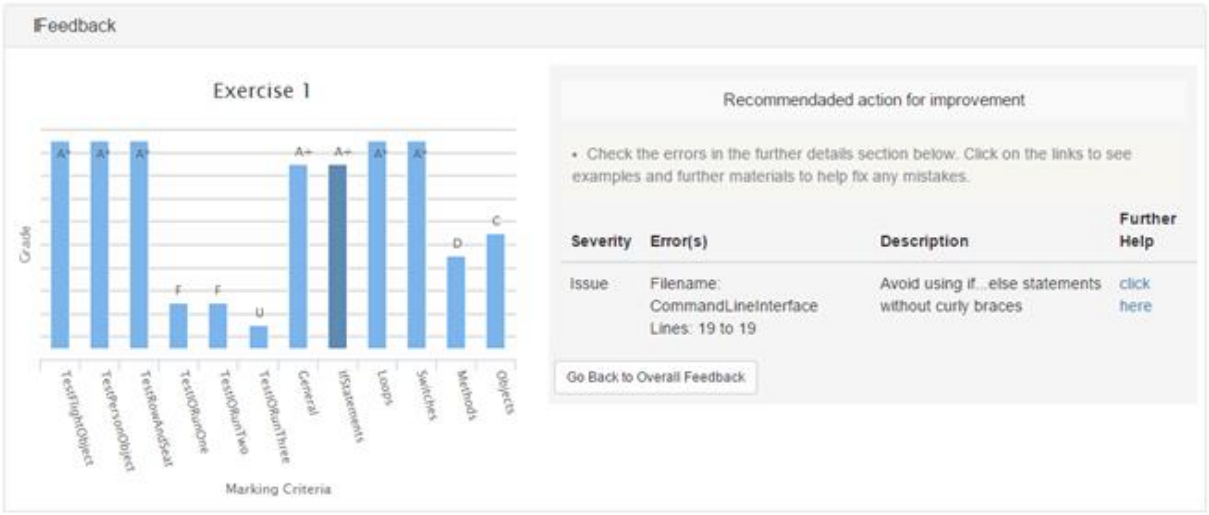

Figure 1. Example exercise showing the if-statements part selected. The student may click on the bar chart to view issues related to any section.

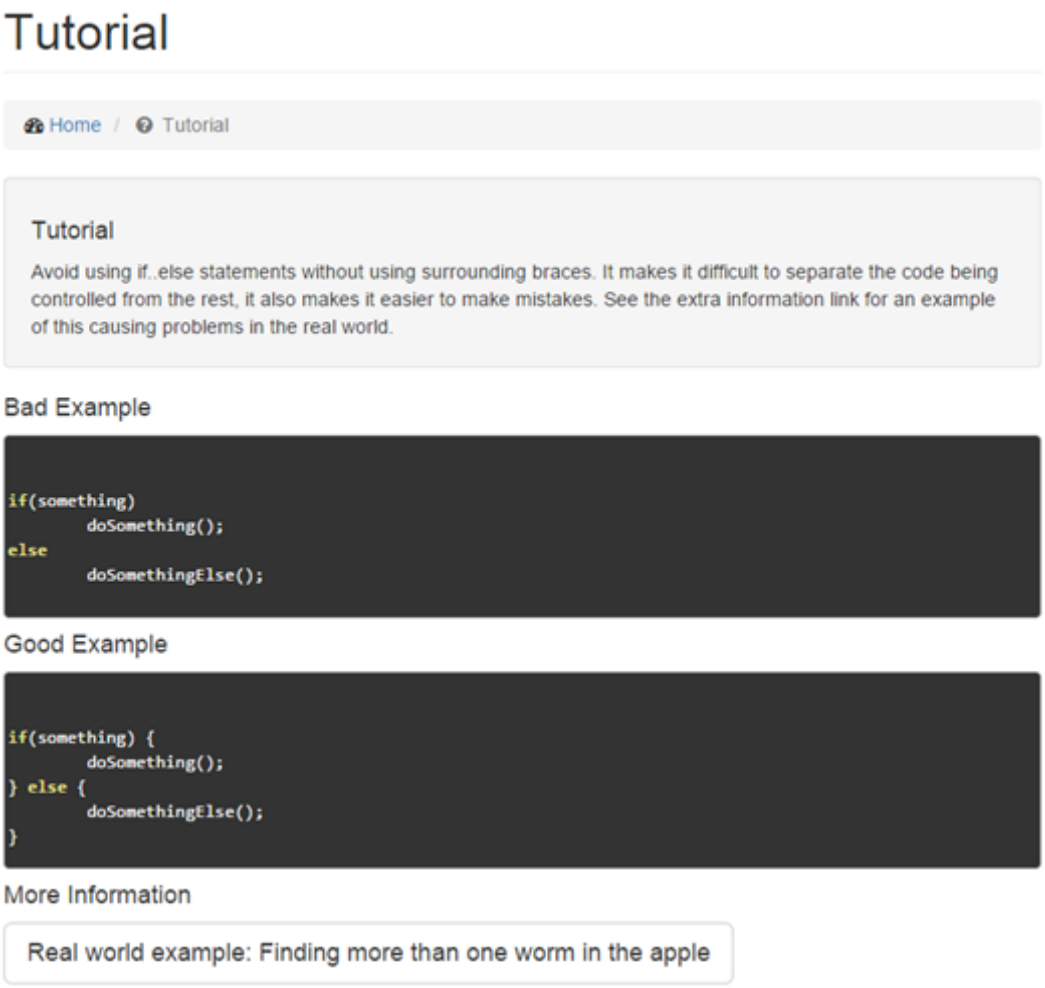

Figure 2. A short tutorial for one common mistake, showing a short description, example and with a link to external resources for further reading (if required).

This example (Table 1.) is trivial in nature (for illustration purposes), there are only a few lines of code and the problems have appeared close together. For a real coursework submission, in particular the ones used for this study, there are approximately 10 files per submission, each of which may have hundreds of lines of source code with many different rule violations.

The feedback provided in this way meets the good practise requirements discussed earlier in the criteria for high quality feedback. The criteria for the assessment is clear, and the results are specific, timely, legible and clear. The feedback, including the chart and break down of rule violations is presented to students instantly. This allows students to identify and fix their submission, with the use of the examples if multiple submissions have been made available. Additionally, the examples are purposely not specific to a single exercise and therefore some effort is required on the part of the student to consider how the example applies to their own work. Furthermore, for the module convenor, statistics on the results can be easily collected to find rules that the cohort as a whole are commonly violating, making it possible to tailor lectures to the cohorts needs. If the same work is re-submitted, the resulting feedback and grade will be identical, thus fulfilling the criteria for consistency and reliability. Laboratory help sessions are also set up for students to openly ask questions regarding their feedback to facilitate discussion between staff and students.

Handley et al. (2007) highlight the advantages of finding time to give feedback on drafts. The main premise is that students are able to submit their work and obtain some feedback which can be applied, and can learn from the process of doing so. Case studies support the assumption that students learn from applying feedback, however they also illustrate the potential time-cost problems associated with feedback on drafts. With an automated system, this cost is reduced and therefore it is possible to allow ‘pre-submissions’, or assessed drafts. However, allowing multiple submissions raises a concern that students may continue to mechanically tweak the work to gain a slightly improved grade without understanding why. To address this, it is possible to limit the pre-submission assessment to utilise only a subset of the tests used in a final submission; this is particularly useful for assignments that require some manual assessment as the students can have instant partial feedback on drafts before submitting the final version which may also have manually marked parts.

TMA presents feedback instantly upon submission of the assignment. At this point the coursework submission is fresh in the student’s mind. This approach guarantees that all students will be provided with feedback rather than having to actively collect it from an email, office or otherwise. This is considered to be a contributing factor in promoting student engagement with the feedback.

Measuring improvements following feedback

There were 346 submissions, from 60 unique students to the first exercise (Coursework 0). This exercise was not compulsory and multiple submissions were allowed. Although the nature of this assessment was formative, a summative grade was also provided for each part of the marking criteria. The grade did not count towards the students’ final grade for the module. The coursework involved creating a simple command based game in Java. The game involved reading stories from text files and presenting them to the user. The user is then given a set of options (from the text file) to select from. Depending on the options typed in by the user, different text files are read for the following parts of the story. The user goes through the game making choices until they reach the end. The implementation of this coursework involved creating several classes, using if-statements, loops, methods, objects, custom exceptions and some string manipulation. For each component, a ruleset was created from the 127 rules to measure the quality of the source code. Before applying the rules, TMA first checks if the construct being tested exists in the solution. For example, a solution that does not have if-statements, will not be awarded marks for not making common mistakes

relating to if-statements (In this exercise, a sensible solution without if-statements would not be possible).

For each ruleset, the average grade from the first submission and the average grade from the last submission was calculated, allowing overall improvements made by the cohort to be measured. The final column in table 2 excludes those who achieved a perfect score for a ruleset on the first submission as these students have no room for improvement on those particular tests.

Table 2. Average results for each ruleset from the first and last submission of each student.

Ruleset Name	Average first submission mark (%)	Average final Submission mark (%)	Raw Improvement (%)	Improvement excluding perfect grades (%)
Classes	58.1	84.87	26.77	35.51
General	64.03	81.13	17.1	33.54
If-Statements	50.07	73.41	23.34	23.34
Loops	69.23	82.69	13.46	31.25
Methods	29.52	45.70	16.18	19.57
Objects	37.82	66.67	28.85	30.95
Strings	85.53	86.03	0.5	4.65
Exceptions	91.8	97.10	5.3	18
Averages	**60.8**	**77.2**	**16.4**	**24.6**

A substantial improvement is seen from the average grade in the first submission to the average grade in the final submission. This indicates that improvements have been made to the submissions, but not necessarily that learning has taken place. It is possible that the students acted on the feedback by fixing the issues highlighted, but without understanding the reason or being able to identify the issue themselves in future. However, these results indicate that the feedback is clear in identifying the position and nature of these mistakes. It also indicates that the students are engaging with feedback and not just the grade, as they are able to identify and fix the specific problems highlighted, with the vast majority making multiple submissions (on average 5.7 each). These results show a strong indication that the feedback has helped to clarify issues that the student may not have previously understood or been aware of. This partially addresses the concerns highlighted in the low scoring NSS results for the statement 'feedback on my work has helped me clarify things I did not understand'.

However, these results do not prove deep learning has taken place, they only show that when an issue is explained and highlighted, the students are able to understand and fix it. The results are validated by a closing student survey, and then by comparing student performance in a subsequent assignment to a previous cohort, we are able to assess the impact on learning of this technique.

Survey of students following the first assessment

The aim of the survey was to determine if students feel this type of feedback and assessment technique has assisted in their learning process. Some questions included on the survey are not relevant to this paper and therefore have been omitted. There were 35 responses to the survey; not all questions were compulsory and some respondents chose to skip them.

1. *Did you read the instant feedback that was provided?*
    a. Yes: 22 (100%) No: 0 (0%)
2. *Did the feedback highlight areas of your work which could be improved?*
    a. Yes: 20 (91%) No: 2 (9%)
3. *Did the short tutorials showing the good code example and the bad code example help you to understand how to improve parts of your work that were marked as problems?*
4. Yes: 15 (68%) No: 7(32%)
5. *Do you feel you improved the quality of your submission based on the feedback?*
    a. Yes: 17 (77%) No: 5(23%)
6. Do you feel the feedback assisted in your learning, and will help you in future work?
    a. Yes: 15 (68%) No: 7(32%)

These responses indicate that students strongly agree that the feedback technique highlighted parts of the work which could be improved. Due to the nature of the technique, this is to be expected. Generally, students felt they were able to improve the quality of their submission based on the feedback provided and that the feedback assisted in learning.

Measuring the impact on future assignments

The previous study compared first/last submissions from the same cohort to a single exercise, showing that students improved their submissions following feedback comments. This section discusses subsequent work submitted by same students to the next exercise, after benefitting from the feedback produced by TMA. This part of the paper examines if the type of feedback they were provided was effective in terms of 'feeding forwards' into the following exercise and tests if students did learn and apply this to the next exercise. This part tests if the students have gained self-efficacy and are able to apply the lessons learnt from part one, without relying on TMA to identify common mistakes for them. Part one allowed for multiple submissions, so all of the students have benefitted the process of receiving detailed feedback, fixing their mistakes and resubmitting. Measurements of student performance in the subsequent exercise (part two) are compared to results from a previous cohort who had not benefitted from the high quality feedback or this assessment process. This allows an investigation into the quality of work submitted by this cohort, compared to that of a previous cohort whose work was manually assessed and did not meet the criteria for good quality feedback (due to time constraints and the consistency of manual assessment).

The students undertook the main assignment (coursework one), which was to create a HTTP 1.0 server using Java. This exercise has been used in previous years (allowing comparison), and is split in three parts.

Part one involved creating the networking classes, dealing with threading (for multiple clients) and creating a minimal implementation of a request handler which returns valid 'HTTP 1.0 not implemented' responses. For this part of the assignment, a single pre-submission was allowed for formative feedback, before the final submission. As this coursework was compulsory, all students completed it. However, for students who had not submitted the optional coursework zero, this was the first experience TMA meaning they had not yet benefitted from the type of feedback produced by TMA. Therefore, part one was not used for the comparison.

Part two, submitted two weeks after part one, involved completing the request handler and implementing classes to deal with reading/writing files to/from the file system to create valid responses. For this part, pre-submissions were not allowed and every student must have completed part one, and therefore they must have had experienced and engaged with (due to allowing multiple submissions on part one) the detailed and instant feedback generated by TMA.

For comparison, we considered the number of rule violations introduced in the development of part two of the assignment. This provides insight as to whether the experience from using TMA has affected future work and learning, as it is possible to compare to a cohort from a previous year whose work was manually assessed. The module convenor, lecture notes and teaching methods remained constant, that only the assessment technique and the type and quality of feedback changed.

In the previous year (2012/13), 118 students submitted part two of the coursework. In the current (2014/2015) year who were assessed using this technique, 141 submissions were made. The difference is due to an increase in the number of students on the course.

Table 3. Table showing differences in violation count for historic submissions (without experience of assessment using the code analyser) and new submissions.

	Without experience of code analyser assessment (2012/13 submissions)	After experience with code analyser assessment (2014/15 submissions)	Difference
Coursework two, part 2, average number of new unique violations	30.38	19.62	10.76 (35.4%)

Discussion of results and conclusions

The results of coursework two strongly support the outcome of the surveys, indicating that the type of feedback provided has assisted in learning. The comparison to the historic cohort shows that the students who have been given precise, detailed instant feedback (in line with the criteria) produced less of the common mistakes they were given feedback on. In previous years, feedback was manually produced, resulting in it being less specific (but relatively unique) and given out several weeks after submission. The workload to assess this work previously involved five markers and took several weeks to process, using this technique it takes several hours to set up the assessment, and no additional time for marking. The students have been able to improve their work based on the feedback provided, and have gone on to produce better work in the following exercise. It is not possible to indicate exactly which of the general good practises of feedback discussed in the introduction caused this effect. It could be due to the timely manner of the feedback, thoroughness, links back to learning resources or consistency. However, the results make it clear that altering the assessment technique and striving to meet feedback best practises has the potential to impact substantially on learning. This study highlights the importance of feedback and the decisions in assessment techniques and shows real differences in student performance following different assessment techniques.

The results also highlight the effectiveness of automated assessment for certain types of work, beyond the typical assumptions that it saves time. The student survey results suggest it's useful and desirable for automated assessment to be utilised to improve feedback turnaround time, but also that it is possible to produce more specific, improved feedback with the help of automation. However, survey results also highlight that students do not see automated assessment as a replacement for manual marking, but that it is preferred to be used in conjunction with manual assessment.

Implications for HE assessment?

Within the more general higher education context, this paper has highlighted the importance of good quality assessment techniques and the impact of high quality feedback on student learning.

Broad criteria for defining good feedback has been collected and applied with positive results. Although the specific rule based technique may not be applicable for all disciplines outside of discipline of Computer Science, the criteria followed for good quality feedback are. Academic staff and teachers should be encouraged to target feedback best practises as a direct method for improving teaching and learning.

Coursework zero and coursework one showed students improving their submissions based on feedback given on drafts (pre-submissions). This approach encourages engagement with the feedback, which as measured on the subsequent exercise, led to improved performance when compared to students who were given feedback on drafts (or feedback with as much detail). These findings highlight the importance of feedback on drafts, and are in agreement with Handley et al. (2007). Therefore, it is recommended for all disciplines, where possible, to utilise any early feedback mechanisms available to encourage engagement with feedback.

The authors are of the view that automated assessment is not a replacement for staff-student interaction, or just a way to save time. However, if applied with care it can be a method for releasing time for interactions such as feedback sessions or help desks. It is a common misconception that efforts in automated assessment are merely attempts to save time (Beesley et al., 2015). Within certain disciplines, automated assessment can provide improvement in consistency and fairness whilst also unlocking the pedagogic advantages of instant feedback, which can have a measurable impact on student learning.

For disciplines outside of Computer Science, a range of e-assessment systems exist for automated or semi-automated assessment, including Numbas for Mathematics (Numbas, 2016), Moodle Quiz for short answers and multiple choice (Moodle, 2016) or CourseMarker for programming and diagram based assessment (Higgins et al, 2009) and many more. This paper highlights that use of these tools, when used in synergy with the feedback criteria outlined in this paper can lead to benefits more than just time saving. Furthermore, for other disciplines, it is noted that automated detection of common mistakes and in turn generation of specific, detailed feedback (with examples) on rule violations is a valid and useful approach. For example, detecting common mistakes in mathematics is possible and tailored feedback can be provided using Numbas or Moodle. As shown in this study, the process of allowing the students to view detailed feedback, fix their submission and re-submit results in better quality submissions not only to the current assignment, but in subsequent assignments. This suggests the approach does promote student learning, rather than a mechanical fixing of issues raised, and use and development of similar tools for other disciplines where possible is recommended.

References

Beesley, S., Nutbrown, S. and Higgins, C. (2014) A framework for facilitating feedback best practises based on a study of staff and students. *HEA STEM Annual Conference*. Edinburgh: University of Edinburgh, 30 April. Available at: https://www.heacademy.ac.uk/resources/detail/stem-conference-2014/Post_event_resources/GEN/Facilitating_feedback (Accessed:15 March 2016).

Beesley, S., Nutbrown, S. and Higgins, C. (2015) Preconceptions surrounding automated assessment - A study of staff and students. Fifth International Assessment in Higher Education Conference. Birmingham: Maple House, 24-25 June 2015.

Bellon, J., Bellon, E. and Blank, M. (1992) *Teaching from a research knowledge base: A Development and Renewal Process*. New York: Merrill.

Black, P. and Wiliam, D. (1998) 'Assessment and Classroom Learning', *Assessment in Education: Principles, Policy & Practice*, *5*(1), pp.7-74.

Duncan, N. (2007)'"Feed-forward": improving students' use of tutors' comments. *Assessment & Evaluation in Higher Education',* 32(3), pp. 271-283.

Handley, K., Millar, J., Price, D., Ujma, D. and Lawrence, L. (2007). When less is more: Students' experiences of assessment feedback. *HEA Annual Conference*. Harrogate: Harrogate International Centre, 3-5 July. Available at: https://www.heacademy.ac.uk/resource/when-less-more-students-experiences-assessment-feedback (Accessed 15 March 2016).

HEFCE (2015) National Student Survey results 2014. Available at: http://www.hefce.ac.uk/lt/nss/results/2014/ (Accessed: 16 April 2015).

Higgins, C., Gray, G., Symeonidis, P. and Tsintsifas, A. (2005) 'Automated assessment and experiences of teaching programming', *Journal on Educational Resources in Computing (JERIC),* 5(3), pp.287-304.

Higgins, C., Bligh, B., Symeonidis, P. and Tsintisfas, A. (2009) 'Authoring diagram-based CBA with CourseMarker', *Computers & Education,* 52(4), pp.749-761.

Higgins, R., Hartley, P. and Skelton, A. (2002) The Conscientious Consumer: Reconsidering the role of assessment feedback in student learning', *Studies in Higher Education*, 27(1), pp.53-64.

Irons, A. (2008) Enhancing learning through formative assessment and feedback. Abingdon: Routledge.

Martin, R. (2008) Clean code: a handbook of agile software craftsmanship. Upper Saddle River, NJ: Prentice Hall.

Moodle (2016) *Question types.* Available at https://docs.moodle.org/30/en/Question_types (Accessed: 17 March 2016).

Nicol, J and Macfarlane-Dick, D. (2006) 'Formative assessment and self-regulated learning: A model and seven principles of good feedback practice', *Studies in higher education*, 31(2), pp. 199-218.

Numbas (2016) Numbas really versatile maths e-assessment. Available at http://www.numbas.org.uk/ (Accessed: 17 March 2016).

PMD (2015) PMD is a source code analyser. Available at: http://pmd.sourceforge.net/ (Accessed 12 May 2015).

Price, M., Handley, K., Millar, J. and O'Donovan, B. (2010) 'Feedback: All That Effort, But What Is The Effect?', *Assessment & Evaluation in Higher Education,* 35(3), pp.277-289.