Issues in Informing Science + Information Technology

An Official Publication of the Informing Science Institute
InformingScience.org

IISIT.org

Volume 16, 2019

COMBINING SUMMATIVE AND FORMATIVE EVALUATION USING AUTOMATED ASSESSMENT

John English * Ramat Gan Academic College, Israel john.e@iac.ac.il

Tammy English Ramat Gan Academic College, Israel tammy.e@iac.ac.il

* Corresponding author

ABSTRACT

Aim/Purpose	Providing both formative and summative assessment that allows students to learn from their mistakes is difficult in large classes. This paper describes an automated assessment system suitable for courses with even 100 or more students.
Background	Assessment is a vital part of any course of study. Ideally students should be given formative assessment with feedback during the course so students and tutors can identify weaknesses and focus on what needs improvement before summative assessment, which results in a grade. This paper describes and automated assessment system that lessens the burden of providing formative assessment in large classes.
Methodology	We used Checkpoint, a web-based automated assessment system, to grade assignments in a number of different computer science courses.
Contribution	The students come from diverse backgrounds, with a wide range of ages, previous qualifications and technical skills, and our approach allows the students to work at their own pace according to their individual needs, submitting their solutions as many times as they wish up to a deadline, using feedback provided by the system to help identify and correct their mistakes before trying again.
Findings	Use of automated assessment allows us to achieve the goals of both summative and formative assessment: we allow students to learn from their mistakes without incurring a penalty, while at the same time awarding them a grade to validate their efforts. The students have an overwhelmingly positive view about our use of automated assessment, and their comments support our views on the assessment process.
Recommendations for Practitioners	Because of the increasing number of students in today's courses, we recommend using automated assessment wherever possible.

Accepting Editor: Eli Cohen | Received: November 19, 2018 | Revised: January 14, 2018 | Accepted: March 30, 2019

Cite as: English, J., & English, T. (2019). Combining summative and formative evaluation using automated assessment. *Issues in Informing Science and Information Technology, 16,* 143-151. https://doi.org/10.28945/4293

Keywords automated assessment, formative and summative evaluation, self-paced learning, feedback, 'little and often' assessment

INTRODUCTION

Assessment is a vital part of any course of study. It is primarily seen as a way of validating that the student has mastered the course material. *Summative assessment* enables tutors to measure the student's performance against some standard benchmark. End-of-course examinations are used to determine whether or not a student has met the learning objectives at the end of a course, while in-course assignments are used to measure student progress prior to the end of the course. These assignments are handed in by specified deadlines to be graded before being returned to the student with comments from the tutor.

By contrast, *formative assessment* is assessment which does not result in a grade affecting the outcome of the course. Instead, it helps students and tutors identify learning deficiencies, both for the individual as well as for the group as a whole. This can help focus a student's attention onto topics which are imperfectly understood, and it can also provide the tutor with feedback which can be used to modify subsequent learning activities. This can help clarify topics which are poorly understood by the class as a whole. This is particularly important when dealing with diverse groups of students, as we do: students with widely varying ages, social and educational backgrounds and technical skills.

The ideal assessment would combine both approaches: one which is summative, but which includes a feedback loop of criticism and improvement. For example, a student might present a piece of work in a tutorial setting for a formative evaluation, and then subsequently rework it in the light of the feedback received before submitting a final version for summative grading by the tutor.

However, large class sizes have become increasingly common in higher education over recent years. Lecture classes of over 100 students are not unusual, which poses a problem for staff involved in assessing in-course work if grading is done manually. It can be a formidable task to grade 100 pieces of work and provide useful feedback to each student. If the workload is to be kept to a manageable level, the amount of work submitted by students must be minimised or more staff must be involved in the grading process. Feedback to students is often delayed as a result of the time spent grading their work, and where large numbers of staff are involved there can be inconsistencies in the standards applied when awarding grades or providing feedback.

In this paper, we describe our experience of using automated assessment on five computer science courses taught at the authors' institution, and how this experience has affected the way we think about the assessment process. The courses we describe here belong to a B.Sc. programme in Information Systems Management. The programme is intended to equip students with the skills needed for jobs in modern high-tech environments and provides a more rounded and holistic experience than traditional degrees in subjects like Computer Science. It involves roughly equal measures of mathematics (including statistics and economics), computer science (including object-oriented programming and web application development), and business management skills. The emphasis is on solution building, and the course is rounded off by a final project which involves investigating innovative solutions to real problems faced by local companies.

The students are generally mature students in their late 20s or early 30s, many of whom already have jobs in the high-tech sector but are looking for opportunities for career advancement. Many of them also have families and children. The classes are, therefore, held on three evenings each week as well as on Friday mornings to accommodate their needs, and the students need to be highly motivated to succeed. Some already have degrees in other subjects, while others have no previous formal qualifications. Other students are progressing in a conventional manner from school matriculation to a degree course. Some students have a technical background, others do not. Because of this diversity within the student cohorts, assessment outcomes can vary widely. We encourage the students to work in groups where stronger students can assist the weaker, but the only way the weaker students can really

learn is to make sure they have sufficient practical work to ensure that they can properly assimilate the skills being taught.

By using automated assessment, we are able to assess student progress using a 'little and often' pattern, where the students produce many small pieces of work during the course of their studies. It is especially valuable in subjects like ours where students need to learn from practice; and in particular, to learn from their own mistakes. In these contexts it allows students to perform as much practical work as possible in the time available. Students benefit from instant feedback, consistent and impartial grading, and the ability to progress at their own pace. They will thus be able to gain more from assessment as part of the learning process as a whole. Staff benefit from a reduced marking workload, but they can also use information from the automated marking system to focus on student weaknesses and misconceptions and thus improve their teaching.

THE GOALS OF ASSESSMENT

As a result of our experiences, we have come to regard assessment in a new light. The conventional approach to assessing students typically involves setting assignments during the course, followed by a written end-of-course examination. The assignments are intended not just as a quantitative measure of student progress, but also (in theory) to provide students with timely feedback that they can use to identify their own weaknesses and remedy them in time for the exam. Staff can also use assignments to identify and correct misconceptions on the part of the students. Unfortunately this idealistic view of the assessment process is only possible in rare cases. For example, in the Oxford tutorial model (Palfreyman, 2008) the students meet individually or in small groups with a tutor each week. The tutor then sets work based on the needs of the individual students which is then presented and discussed the following week.

In most institutions today large class sizes are the norm, and the student/staff ratio means that the Oxford tutorial model is unfortunately impractical. The time available for staff to grade assignments is severely limited, which means that there will normally only be one or at most two assignments per semester. The assignments are typically set with deadlines towards the end of the semester, with grades made available only just before the final exam. This means that the students have little or no feedback on their progress during most of the course, and the feedback they do receive may be too late for learning difficulties to be resolved in time for the exam. And of course there is no opportunity to put things right, to learn from the mistakes that were made and correct and resubmit the work. One of the best ways to learn is to learn from the mistakes we make, but this requires the opportunity to identify and correct those mistakes: to try again, in other words.

One solution to this is to provide additional formative assessment tasks which can give the students more practice in the skills needed for the summative assignments and exams. The problem here is that again, staff have little time to grade these exercises and provide feedback. Another problem is that less-motivated students will not bother with unassessed formative exercises because they do not contribute directly to the final grade for the course.

Sometimes formative exercises are provided during 'tutorial' or 'lab' classes, with the tutor watching how the students tackle problems and advising them how to progress, before revealing the solutions later. However, this approach has problems with classes from diverse backgrounds, with a range of knowledge, ability, and maturity levels, by preventing students from progressing at their own pace. Good students may find the exercises insufficiently challenging, and may, therefore, become bored, while weak students might find they are unable to complete the exercises in the time available and become disillusioned as a result. All of this can have a serious impact on motivation, and thus on attendance and engagement.

Automated assessment, where work is graded automatically by computer, provides a solution to these problems. In topics such as computing and mathematics, it is relatively easy to assess 'correctness' along different dimensions; in other subjects (e.g., literature) this is not currently feasible, although

some preliminary work has been carried out and advances are being made (Shermis, 2014). For example, in a computer science programming course it is possible to automatically determine answers to questions such as:

a) Does this piece of program code compile?
b) Does this piece of code produce the correct output results for various inputs?
c) Is this piece of code efficient in terms of execution time for different inputs?
d) Does this piece of code use some required approach (e.g., is it a recursive solution)?
e) Is this piece of code elegant according to some measure (e.g., number of lines of code)?

By summing correctness across a number of different axes, it is possible to provide more nuanced assessments than a simple 'right' or 'wrong' decision.

Automated assessment solves the problem of staff time for grading. It also ensures that grading is impartial and consistent. Because grading is effectively instantaneous, assessment can be carried out on a 'little and often' basis, with assignments every week or two throughout the semester. Students can work at their own pace, subject only to the final submission deadlines imposed by the grading system. It also makes it feasible to allow students to correct any mistakes they made and resubmit the work prior to the deadline, thus giving the benefits of formative assessment within the framework of a summative assignment. This of course means that the assessment system must provide detailed feedback on how and why the grades were awarded as they were so that students can learn from their mistakes and correct any misconceptions they may have.

AUTOMATED ASSESSMENT WITH CHECKPOINT

We have been using automated assessment since 2013 on five computer science courses on our degree programme:

- Introduction to Programming in C#
- Object Oriented Programming in Java
- Development, Design and Management of Databases
- Algorithms and Data Structures
- Computer Architecture and Operating Systems

In-course assessment worth 30% of the total grade is conducted using Checkpoint (English, 2006), a web-based automated assessment system developed by the first author, which has been used by the authors at three different institutions in the UK and Israel since 2005. The remaining 70% of the grade comes from a traditional end-of-course written examination. The passing grade for the course is 60%, so there is an incentive to complete the assignments; however, an exam grade of at least 60% is also required, so a good assignment grade does not make it any easier to pass the course. There are normally three or four assignments for each course, where each assignment consists of between 5 and 12 questions of varying difficulty and automatically-enforced submission deadlines that are two or three weeks apart. For the more advanced students, there are normally extra formative assignments (that is, the assignment grade has a weighting of zero when used to calculate the final grade) so that those who finish the assignments early have the option to do something extra to stretch themselves a little.

Checkpoint supports many different question types, including conventional *fixed-response* (e.g., multiple choice) questions, where the student has to choose the correct answer(s) from a list. However, in the courses described here, the assignment questions are all of the *free-text* variety; that is, a question provides one or more text entry fields for the student to type in an answer. Figure 1 shows a sample question from a programming course. The submitted answer is embedded in a test program, compiled and then executed using test data which is usually randomly generated within certain parameters. The grade is displayed immediately, together with automatically-generated feedback.

Checkpoint allows a variety of optional restrictions on submissions, including a limit on the total number of attempts, a final deadline after which submissions will no longer be accepted, a cut-off date after which reduced grades will be awarded, and a limit on the number of attempts which can be submitted in any 24-hour period. However, use of these restrictions in the past did not show any significant benefit from their use and served only to increase the pressure on the students, so on our courses we decided to allow the students to submit an unlimited number of attempts for each exercise up to the deadline, which also means they do not have to complete an entire exercise at one sitting. The effect of this is that students normally answer one question at a time, correcting the answer until they get it right, and only then moving on to the next one, working at their own pace.

Lab classes are often used as Checkpoint 'help sessions', where the students can get individual assistance from the tutor with a particular problem, and they can work collaboratively to understand and solve their problems. To deter plagiarism, the questions are randomized in various ways, so that the students cannot just copy each other's' answers. Questions within an exercise can be shuffled into a random order. Question groups can be defined, where a specified number of questions will be chosen at random from a larger set of available questions. Questions can also be parameterised by inserting one or more items chosen at random from a set of possible values. For example, questions involving numerical data can be individualized by making random choices from specific ranges of values. Each student thus has an individual set of similar but not identical questions.

Typically, at the end of the course, most of the students have managed to achieve full marks for the assignments. This means that the assignment results do not serve to differentiate between students by ability; however, we feel that the learning that results from correcting answers and resubmitting is the primary goal of the in-course assessment process, and we leave it to the final exam, with its 60% pass mark, to provide differentiated outcomes.

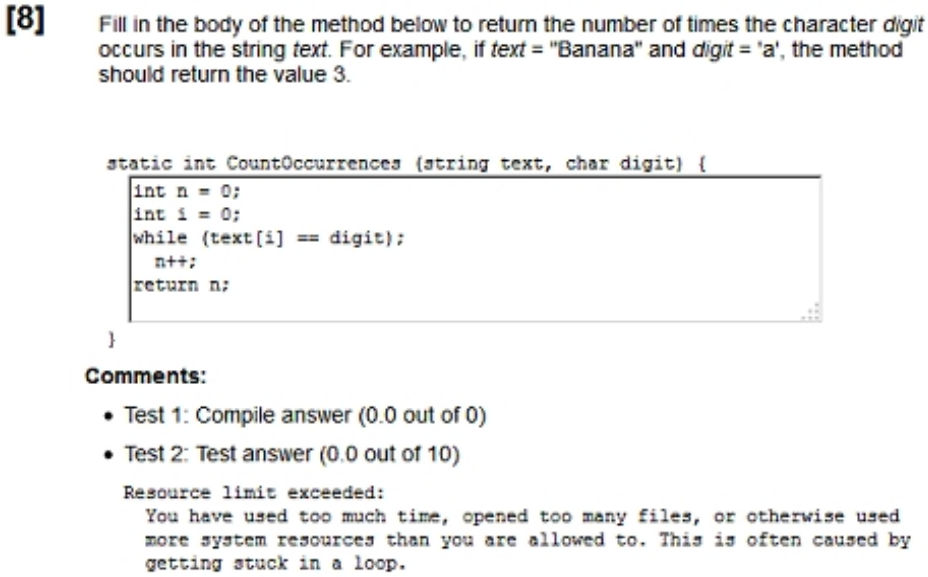

Figure 1: Sample question from a programming course, with feedback

From a staff viewpoint, one of the benefits of an automated assessment system is the ability to track student progress, particularly when a 'little and often' assessment regime is used. Checkpoint provides reporting facilities which allow tutors to 'drill down' to any desired level of detail: a table of results for the whole course, results for a particular exercise for all students, results for a particular student for all exercises, results for each attempt by an individual student for a particular exercise, and ultimately a copy of an individual attempt for an exercise by a particular student. There are also tables

which give the number of attempts for individual questions for either the whole cohort or for a particular student, which helps tutors identify which questions are causing the most difficulty.

Checkpoint's grading is done by embedding the solutions into one or more files and then executing a series of *marking scripts* to evaluate them according to any desired criteria. Each script consists of a sequence of commands that can be used to process the files in any desired way to generate an associated number of marks. Several different dimensions of 'correctness' can thus be measured independently, and the results can be combined to give a more nuanced assessment than a simple 'right or wrong' approach (Rosenthal & Suppes, 2013). Checkpoint does not impose any limitations on how correctness is measured; it simply acts as a framework to allow free-text answers to be submitted and tested. Apart from behavioural correctness, answers can be judged on other factors such as style or efficiency. For example, Figure 2 shows a programming question which uses four marking scripts to check whether the submission compiles, whether it works with some randomly-chosen test data, whether the solution is an efficient one, and whether it uses a particular language feature. Anything that we can devise a way to measure can be used for automatic assessment.

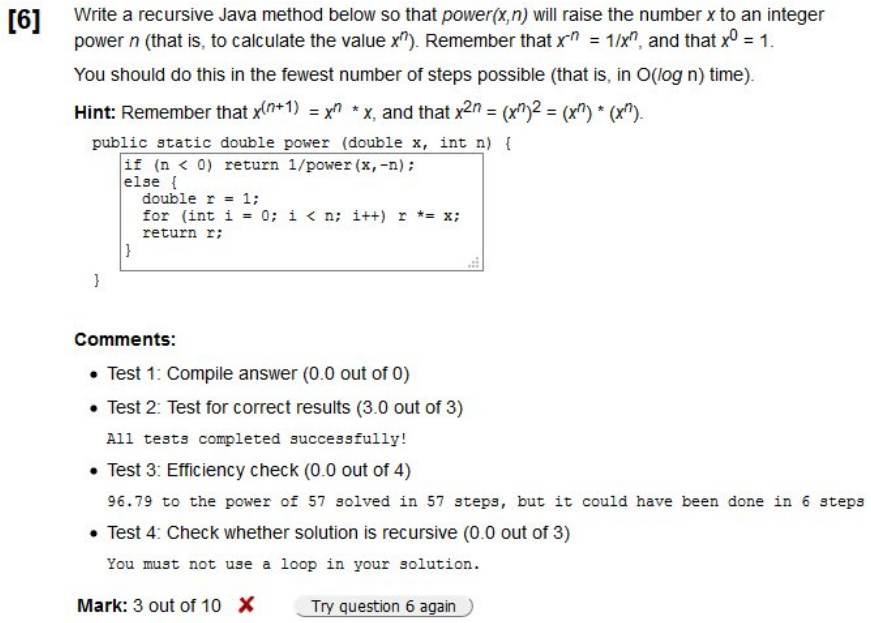

Figure 2: Sample question from a programming course, with an efficiency test

The approach to assessment described here depends crucially on the quality of the feedback generated by the system (Keuning, Jeuring, & Heeren, 2018). The marking scripts not only generate marks, but also output, which will be displayed as feedback to the student. It is, therefore, up to the author of the question to determine what feedback should be given in particular cases. This means that when creating new questions, authors need to put considerable effort into identifying common errors and provide helpful feedback when they are detected. This is not trivial; creating questions that provide accurate feedback for different mistakes takes a great deal of effort, much more so than in a manually-assessed assignment, and questions normally need further refining to improve the feedback in the light of practical experience. However, once good questions have been developed, they can be stored in a 'question bank' where they can be reused as-is or used as the basis for developing new questions.

STUDENT REACTIONS

A detailed empirical analysis of student reactions to Checkpoint was already published in an earlier paper (English & English, 2015) to which interested readers should refer. It described the results of a survey of 141 students on four of the computer science courses described above. The questionnaire comprised 15 Likert scale responses, with values from 1 (strongly agree) to 5 (strongly disagree), as well as five open questions inviting more general ('free-text') responses. The students' reaction to Checkpoint was overwhelmingly positive. In particular, the proposition 'I liked being able to submit multiple attempts' produced an average response of 1.06, or near-unanimous strong agreement.

The responses to the open questions tended to emphasize the benefits of multiple attempts, learning from mistakes using the feedback provided, the ability to work at their own pace from anywhere, and the overall motivating effect of Checkpoint assignments. Several students noted that when an attempt failed, Checkpoint provided feedback that helped them to identify the problem and allowed them to resubmit. This is something that is very hard to achieve if marking is done manually. Checkpoint is evaluating each answer against a comprehensive list of criteria and providing feedback as it does so; a human marker would find it very difficult to be as thorough as this for even a single submission.

From the authors' point of view, Checkpoint's effect on student engagement is quite dramatic compared to similar courses that the authors have taught in the past using a manual assessment regime. Plagiarism is drastically reduced, partly due to question randomization, which ensures that the students do not all get exactly the same questions as each other. However, it is also due to the fact that the pressure to get the answers right the first time has been removed, so that students can relax and work at their own pace, correcting mistakes and trying again. Checkpoint provides detailed feedback that tells them what has happened to lose them marks, and they then have to discover why and correct it. The assignments thus become a personal struggle between the student and Checkpoint, where to copy an answer from someone else would seem like an admission of defeat.

CONCLUSIONS

We strongly believe that assessment should not merely be concerned with the measurement of attainment, but should encourage and assist the learning process. The goal of assessment should not be purely summative, where a student is graded on his or her understanding of the course material at a particular moment in time (and conversely penalized for a lack of understanding). It should also be formative, where feedback is given to help the student identify misunderstandings and to improve their grasp of the material. Feedback provided as part of a summative assessment comes too late to help the student improve (since the grade has already been awarded), while students often tend to avoid formative assignments that increase their workload without any direct reward in terms of their grades.

The ideal solution is summative assessment that provides feedback to help students improve, but where resubmission is possible to demonstrate that there has in fact been an improvement. In a manual assessment regime, the grading workload normally renders this option impractical. However, in an automated assessment system it is a perfectly natural approach. We have demonstrated that by using Checkpoint to assess our courses, we have achieved the goals of both summative and formative assessment: by combining the two approaches we have allowed students to learn from their mistakes without incurring a penalty, while at the same time awarding a grade to validate their efforts.

Students have expressed satisfaction with automated assessment in the past (Rosenthal, Suppes, & Ben-Zvi, 2013), feeling that the instant feedback that it provides, as well as consistency of marking, is an advantage over manual marking. The results of our earlier survey of the courses discussed in this paper (English & English 2015) certainly reinforce that view. Student feedback on using Checkpoint was very positive, emphasizing the benefits of multiple attempts and a quick turnaround time for

submissions, as well as the ability to work at their own pace and in their own time. Many students also said that Checkpoint gave them confidence in learning and the motivation to practise. This feedback and resubmission cycle is something that would be impossible to achieve in a manual grading system. It also illustrates how learning best takes place: by being given the opportunity to make mistakes and then to learn from them and correct them.

The tutors are also satisfied, not only because of the reduction in the time spent marking, but also because of the ability to monitor student progress in detail week by week throughout the course. Students do not have identical questions, which reduces plagiarism, but in addition they do not feel that they have to get a correct answer the first time they try, and there is a greater sense of engagement and achievement as they can tackle their problems one step at a time.

Checkpoint is a versatile assessment framework that allows students to learn from and correct their own mistakes. Students are very positive about the beneficial effect of this on their learning experience. They can progress according to their own pace, and the tutor can track student progress in real time, either as a group or individually, to identify common mistakes or misconceptions, as well as to see what the 'hard' topics are based on the information that Checkpoint provides. Unlike conventional assessment, most students get full marks for their assignments in the end, but to achieve this they put in an enormous amount of effort, and they learn a great deal by doing so. And in our opinion, this is surely the primary purpose of assessment: to encourage learning and to validate it, not to penalize students for a lack of learning while the course is still in progress.

REFERENCES

English, J. (2006). The Checkpoint automated assessment system. In T. Reeves & S. Yamashita (Eds.), *Proceedings of E-Learn: World Conference on E-Learning in Corporate, Government, Healthcare, and Higher Education 2006*, pp. 2780–2787.

English, J., & English, T. (2015). Experiences of using automated assessment in computer science courses. *Journal of Information Technology Education: Innovations in Practice*, 14, pp. 237–254. https://doi.org/10.28945/2304

Keuning H., Jeuring J., & Heeren, B. (2018). A systematic literature review of automated feedback generation for programming exercises. *ACM Transactions on Computing Education, 19*(1), 3. https://doi.org/10.1145/3231711

Palfreyman, D. (2008) *The Oxford tutorial*. The Oxford Centre for Higher Education Policy Studies. Retrieved from http://citeseerx.ist.psu.edu/viewdoc/download?doi=10.1.1.739.202&rep=rep1&type=pdf

Rosenthal, T., & Suppes, P. (2013). Gifted students' individual differences in a computer-based C programming course. In P. Suppes (Ed.), *Individual differences in online computer-based learning: Gifted and other population,* pp. 253-296. Center for the Study of Language and Information, The University of Chicago Press Books.

Rosenthal, T., Suppes, P., & Ben-Zvi, N. (2013). Automated evaluation methods with attention to individual differences: A study of a computer-based course in C. In P. Suppes (Ed.), *Individual differences in online computer-based learning: Gifted and other populations*, pp. 239-252. Center for the Study of Language and Information, University of Chicago Press Books. https://doi.org/10.1109/fie.2002.1157893

Shermis, Mark. D. (2014). State-of-the-art automated essay scoring: Competition, results, and future directions from a United States demonstration. *Assessing Writing*, *20*, 53-76. https://doi.org/10.1016/j.asw.2013.04.001

BIOGRAPHIES

John English was a senior lecturer in Computer Science for 26 years at the University of Brighton, UK, where he obtained a PhD in Computer Science Education. During this time he published numerous papers in the field of computer science education as well as two textbooks ('Ada 95: the Craft of Object Oriented Programming' and 'Introduction to Operating Systems: Behind the Desktop'). He moved to Israel in 2011, and since 2014 he has been a part-time lecturer at Ramat Gan Academic College. His current research interests are mainly in the field of automated assessment.

Tammy English (formerly Tammy Rosenthal) worked in the USA for seven years, developing multimedia online Computer Science courses for the Education Program for Gifted Youth at Stanford University and received her PhD in Computer Science Education from the Hebrew University, Israel, as a joint project between the two institutions. She continued to develop courses for Stanford following her return to Israel, and is currently a lecturer at Ramat Gan Academic College, teaching Computer Science courses. Her primary research interests are the development of automated evaluation methods for computer programs and developing distance learning and e-learning courses.