Journal of Information Technology Education: Innovations in Practice Volume 14, 2015

Cite as: English, J., & English, T. (2015). Experiences of using automated assessment in computer science courses. *Journal of Information Technology Education: Innovations in Practice, 14,* 237-254. Retrieved from http://www.jite.org/documents/Vol14/JITEv14IIPp237-254English1997.pdf

Experiences of Using Automated Assessment in Computer Science Courses

John English & Tammy English
Israel Academic College, Ramat Gan, Israel

john.foreign@gmail.com; tammy.ros@gmail.com

Abstract

In this paper we discuss the use of automated assessment in a variety of computer science courses that have been taught at Israel Academic College by the authors. The course assignments were assessed entirely automatically using Checkpoint, a web-based automated assessment framework. The assignments all used free-text questions (where the students type in their own answers). Students were allowed to correct errors based on feedback provided by the system and resubmit their answers. A total of 141 students were surveyed to assess their opinions of this approach, and we analysed their responses. Analysis of the questionnaire showed a low correlation between questions, indicating the statistical independence of the individual questions. As a whole, student feedback on using Checkpoint was very positive, emphasizing the benefits of multiple attempts, impartial marking, and a quick turnaround time for submissions. Many students said that Checkpoint gave them confidence in learning and motivation to practise. Students also said that the detailed feedback that Checkpoint generated when their programs failed helped them understand their mistakes and how to correct them.

Keywords: automated assessment, self-paced learning, 'little and often' assessment, feedback, multiple attempts, plagiarism, survey analysis

Introduction

Large class sizes have become increasingly common in higher education over recent years. This poses a problem for staff involved in assessing student work if marking is done manually. 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 marking process. Feedback to students is often delayed as a result of the time spent marking their work, and where large numbers of staff are involved there can be inconsistencies in the standards applied when awarding marks.

A number of surveys show that automated assessment systems of various kinds are increasingly used to overcome these problems. Many automated assessment systems have been described in the literature; for example, Ceilidh (Foxley, Tsintsifas, Higgins, & Symeonidis, 1999), CourseMarker (Higgins, Symeonidis, & Tsintsifas, 2003), TRAKLA2 (Laakso, Salakoski, Korhonen, & Malmi, 2004) and RoboProf (Daly, 1999) which have the potential to improve the assessment experience for both students and staff.

Editor: Keith Willoughby
Submitted: June 7, 2015; Revised: August 17, September 25, 2015; Accepted: October 1, 2015

Ala-Mutka (2005) provides a critique of a number of automated assessment initiatives; Carter et al. report on the results of an international teaching staff survey of automated assessment (Carter et al., 2003), and Rosenthal (2004) showed a clear preference for automated assessment in blind tests. Students can benefit from instant feedback, consistent and impartial marking, 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 can 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. As we will show, these benefits also apply to smaller classes.

By using automated assessment, tutors can 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 computer programming or mathematics where students learn from practice and where they can learn from their own mistakes to correct any misconceptions. In these contexts it allows students to perform as much practical work as possible in the time available.

Although the staff perspective is well covered, we were also interested to discover the students' view of automated assessment. Students are of course now accustomed to using technology in all aspects of their everyday lives, so that online assessment fits naturally into the way they work (Dermo & Eyre, 2008). This paper describes students' experiences with automated assessment in a variety of computer science courses taught since 2013 at the Israel Academic College (IAC) in Ramat Gan. The students involved had no previous exposure to automated assessment systems, so they were well placed to compare traditional assessment practices to what was to them a new experience. The courses we describe here are as follows:

- Introduction to Programming in C#
- Analysis and Design of Object Oriented Programming in Java
- Developing, Design and Management of Databases
- Computer Architecture and Operating Systems

Assessment was conducted using Checkpoint, a web-based automated assessment system developed by the first author. The authors have also had experience of using Checkpoint on a variety of courses since October 2005 at the University of Brighton, UK (English, 2006) and the Holon Institute of Technology, Israel (English & Rosenthal, 2009) on topics including C and Java programming, databases, and compiler construction.

Since Checkpoint is web-based, the students can access Checkpoint from wherever they may be via a web browser. Once students log in to the Checkpoint server, they can access the exercises for each course they are registered on. When answers to an exercise are submitted, Checkpoint responds within a few seconds with marks and feedback comments for the questions that were answered. Students can also go back to look at earlier submissions and the corresponding marks and comments.

In the cases we describe, the students are allowed multiple attempts at each exercise, and do not have to complete an entire exercise at one sitting; the questions they provide answers for are marked and the rest are ignored, which allows them to complete an exercise one question at a time if they wish. The only limitation is a final submission date; submissions are permitted until midnight on the specified date, although individuals can be granted extensions to the deadline if necessary.

Figure 1 shows a sample question from a Java programming course. In this case, the question requires the student to complete three separate Java statements. The submitted answers are embed-

ded in a test program and compiled, and then executed using test data which is usually randomly generated. The marks and automatically-generated feedback are displayed immediately, as shown in Figure 1. Where appropriate, the answers can also be judged on other factors such as style or efficiency, as described below.

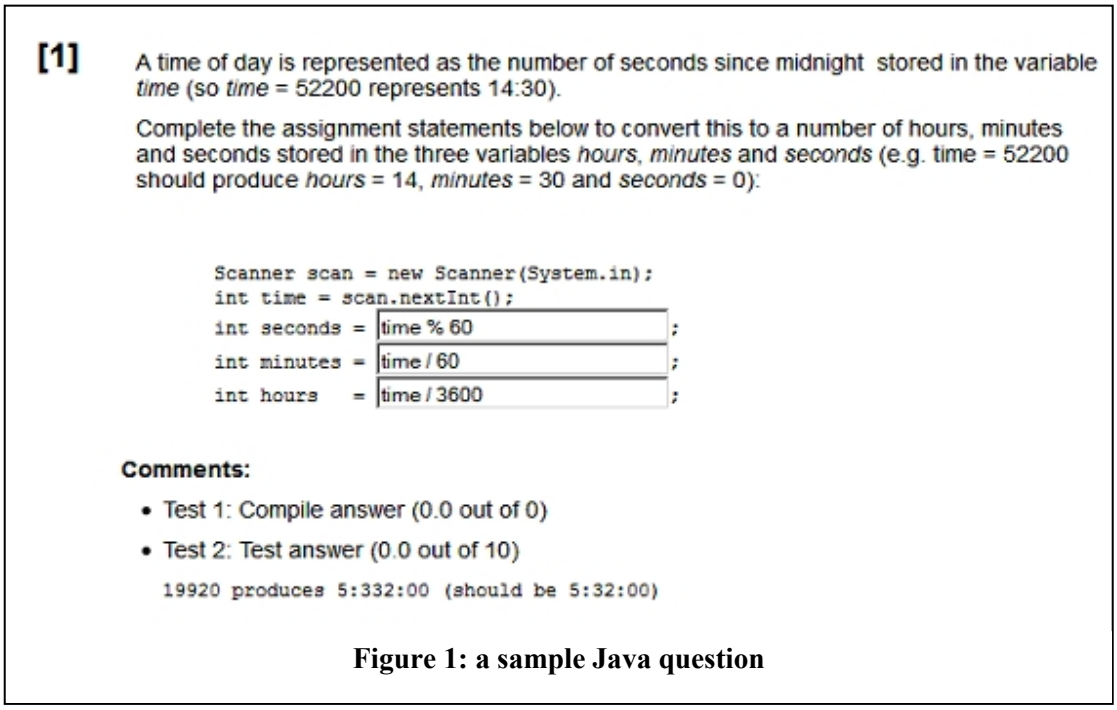


Figure 1: a sample Java question

The IAC Information Systems Management Programme

The courses we describe here belong to an undergraduate degree 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 two evenings each week as well as on Friday mornings to accommodate their needs, and the students need to be highly motivated to succeed.

The authors have taught the courses described in this paper since 2013, using Checkpoint to assess the students' work in all cases. Two other courses (Developing Web-Based Systems and Information Security) also used Checkpoint but the students on these courses were not surveyed. Each of the four courses surveyed had between 20 and 50 students, and the surveys sometimes cover two iterations of the same course.

Assessment at IAC

The courses at IAC all follow a common institution-mandated format, with 4 hours of class contact time per week divided equally between lectures and labs. The assessment weighting in all cases is 30% for assignments and 70% for the exam. The courses described in this paper use Checkpoint for all the assignments.

In the past, the number of assignments per course varied between three and six, typically with 8 to 10 questions per assignment. However, experience showed that setting three assignments (that is, having three separate deadlines) was optimal to avoid overloading the students, who are after all required to submit assignments for other courses in addition to our own. Each assignment now features between 10 and 12 questions. The questions are all *free-text* questions which require the student to type in their answer to the question (which can for example be a fragment of a program or a numerical answer to a problem), and the students can submit an unlimited number of attempts up to the deadline. In some cases students have been known to submit over 100 attempts for an individual assignment, although this seems to be indicative of panic as the deadline approaches rather than a sign of constructive learning!

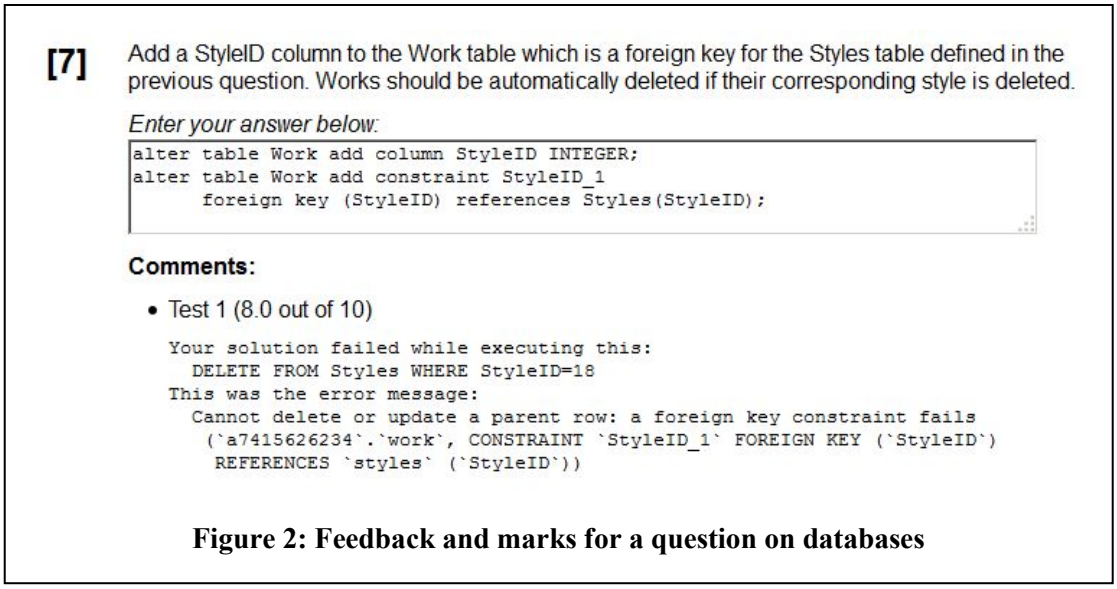
For the more advanced students, there is usually also an extra assignment comprising about 10 more advanced questions (although for the database course, this is an extended exercise involving designing and building a complete database in what is essentially a single question with many parts). The extra assignments have a weighting of zero as far as the assessment for the course is concerned, and are thus used purely as formative assessment. However, there are always several students who are sufficiently enthused to work on this after they have completed the other exercises, while their less able colleagues continue working on the summative assignments.

The fact that the number of attempts is unlimited means that the students are under less stress than they would otherwise be. This leads to more engagement, as they can try to solve an individual problem over and over until they get it right, using Checkpoint's feedback to identify test cases that they may have overlooked in earlier attempts. As a result, there is far less attempted plagiarism than the authors are used to on other courses which are not automatically assessed. Students collaborate on solving problems, but this normally takes the form of explaining to each other how to solve something rather than just providing a solution that others can copy. Also, in order to discourage plagiarism, a certain amount of randomization is used to make it more difficult to copy blindly.

The other aspect of giving students an unlimited number of attempts is that, in all but a few cases, they get full marks for the assignments. However, they are still required to pass the exam, so the assignments are in effect used more as formative assessment than as summative assessment. The fact that they are awarded marks for the assignment that can raise their overall grade (assuming they pass the exam) is an incentive which encourages them to work on the assignments, and once they start working, they usually become hooked on trying to solve the problems. In the opinion of the authors, the important thing is that they *do* the assignments, and with the amount of effort they usually put into completing them, the effect is definitely beneficial in terms of their learning.

Checkpoint

Checkpoint is a web-based online assessment framework which supports a number of different question types. In the courses described here, the questions are all of the *free-text* variety; that is, a question provides one or more text entry fields where the student can enter an answer. For example, on a programming course, the question can ask the student to enter a program or (normally) part of a program to solve a particular problem. Checkpoint responds with a mark and a feedback comment, as shown in Figure 2.


[7] Add a StyleID column to the Work table which is a foreign key for the Styles table defined in the previous question. Works should be automatically deleted if their corresponding style is deleted.

Enter your answer below:

```
alter table Work add column StyleID INTEGER;
alter table Work add constraint StyleID_1
      foreign key (StyleID) references Styles(StyleID);
```

Comments:

- Test 1 (8.0 out of 10)

```
Your solution failed while executing this:
  DELETE FROM Styles WHERE StyleID=18
This was the error message:
  Cannot delete or update a parent row: a foreign key constraint fails
   (`a7415626234`.`work`, CONSTRAINT `StyleID_1` FOREIGN KEY (`StyleID`)
    REFERENCES `styles` (`StyleID`))
```

Figure 2: Feedback and marks for a question on databases

One problem with automated systems is that small errors can be catastrophic (Harris, Adams & Harris, 2004); for example, a missing semicolon in a programming question will mean that the submission cannot be compiled and cannot, therefore, be given any marks by an automated system. Checkpoint allows the tutor to manually 'moderate' the exercise results, adjusting marks where necessary to take care of 'nearly right' answers. Submissions can be anonymised so that student results will only be identified by an institutional registration code to prevent bias where the marks are adjusted manually. An audit trail showing all the changes made to the marks is incorporated in the set of results so that everyone who views the results can see what changes were made and when, who made the changes and why. However, manual intervention is rarely necessary when students are allowed multiple attempts; they can just correct any errors in their solution and resubmit it.

The order and selection of questions in an assessment can be randomized in various ways, as described later. This helps to deter plagiarism by making it possible to present each student with a different set of questions. Where multiple attempts are allowed, the author of the exercise can select whether an individual student will be presented with the same set of questions on each attempt or a new randomly-chosen set of questions. Where the same questions are presented each time, the question is pre-loaded with the previously-submitted solution, ready to be edited and resubmitted. This is generally the best approach for free-text questions, and all the exercises we describe in this paper follow this approach.

Exercise Structure

An exercise consists of a set of questions; these can be divided into discrete sections. A section has a title (e.g., 'Section A') and can specify a maximum number of questions to be attempted; if more than this number are actually submitted, only the best answers up to this number will actually be counted towards the final score. Each section can similarly have a minimum number of answers to be attempted, and this in combination with the maximum limit for the assessment as a whole makes it possible to compose assessments with rubrics along the lines of '*answer five questions, at least one from section A and two from section B*'.

Questions are divided into *fixed-response questions*, where the student is presented with a list of possible answers to choose from, and *free-text questions*, where the answer must be supplied by the student. The traditional multiple-choice question is an example of a fixed-response question,

where the student can choose one of a set of possible answers. As mentioned earlier, the exercises we will discuss in this paper use only free-text questions.

Randomization of various kinds is supported, a feature which can be used to help deter plagiarism. Questions within a section can be shuffled into a random order, and the order of the available answers is normally shuffled in fixed-response questions. Question groups can be defined, where a specified number of questions will be chosen at random from a larger set of available questions. Finally, questions and answers can be parameterised by allowing one or more values to be chosen at random from a set of possible values and inserted into a question or answer. For example, questions in programming courses can use names for functions and their parameters which are chosen at random from a list of plausible names. Where questions involve the use of numerical values, a random choice can be made from a specified range of values.

Free-text questions provide one or more fields for students to enter their answers. Because multiple fields can be provided, this allows for questions in several parts which require separate part-answers. Typically these will be textual answers, but Java applets can also be used to allow submission of non-textual answers such as diagrams. Marking is accomplished by processing the submitted answers using a set of *marking scripts* associated with each question. Each script returns a number of marks to be awarded and can also generate output which will be displayed as feedback for the student.

When a free-text question is submitted, a temporary directory is created on the server for use by the marking process. The question can specify a set of files to be created in this directory, and these can include markers which will be replaced by the values entered into particular fields in the question. This, for example, allows a code fragment submitted by a student in response to a programming question to be embedded into a larger test program. It also allows part-answers to be marked independently of each other or in any desired combination. Once the set of files has been created, a series of marking scripts associated with the question is executed. Different scripts can be used to assess different dimensions of 'correctness'. Each script consists of a sequence of commands which can be used to process the files in any desired way, and an associated number of marks. The commands that make up the script determine what percentage of the available marks should be awarded. The output of individual commands can also be saved in the temporary directory for further processing by subsequent scripts.

To prevent the security risk posed by allowing users to execute arbitrary programs on a mission-critical server, the commands must be chosen from a limited set maintained by the system administrator in a particular directory. These will normally be Unix or Windows shell scripts which invoke other programs elsewhere on the system in a controlled way. In the case of assessing programs written by students, the security risk is even greater, and this can be dealt with using a 'sandbox' system to isolate student programs. On a Linux system this consists of a 'chroot jail' (Schoep, 2005) which isolates the program from the rest of the filesystem and which sets limits on system resources such as CPU time, file space and the amount of output generated. As an example Figure 3 shows an infinite loop being detected in a student's submitted answer. When this happens, Checkpoint is able to generate clear and constructive feedback in order to help the student locate and correct the problem.

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. Anything that we can devise a way to measure can be used for automatic assessment (Whittington & Hunt, 1999); for example, there are several techniques for automatically assessing essay questions (Valenti, Neri, & Cucchiarelli, 2003) which could certainly be used with Checkpoint, although so far this has not been attempted. To date, Checkpoint has been used to assess programming exercises in C, Java, C#, Ada and assembly language, as well as other computer science-related topics such as ASP, SQL,

HTML, Unix shell scripting, binary-to-decimal conversion, logic design, and compiler construction.

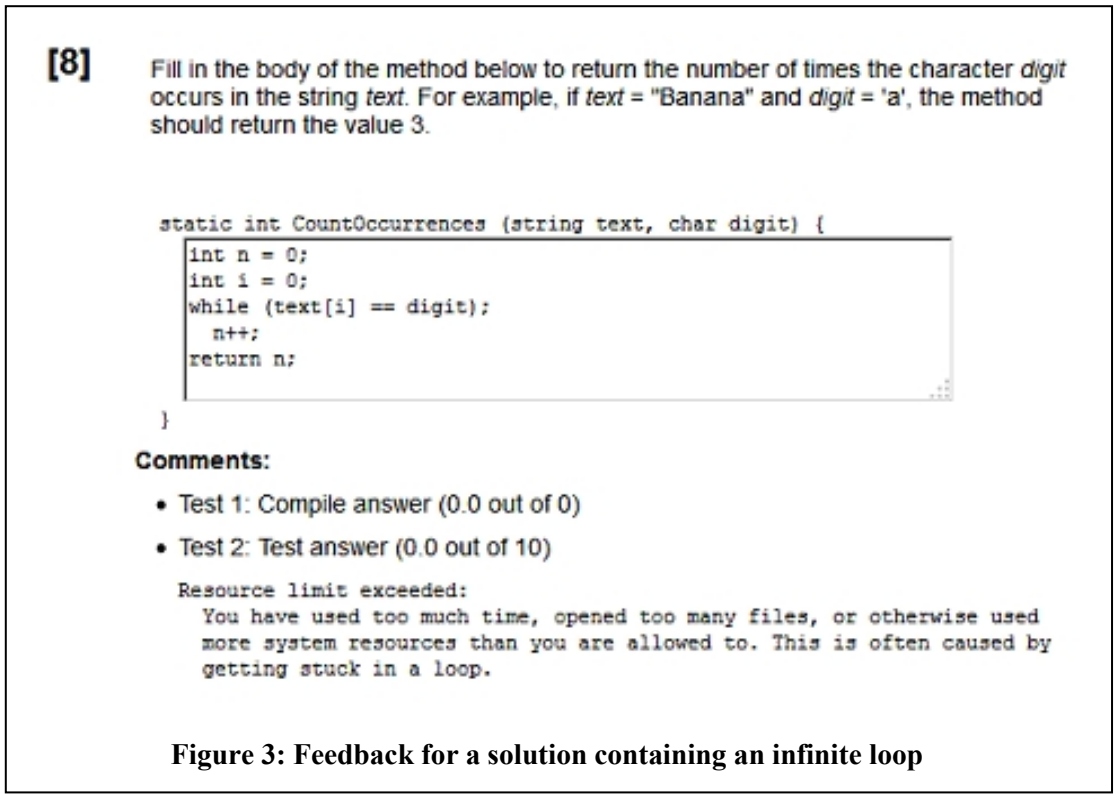
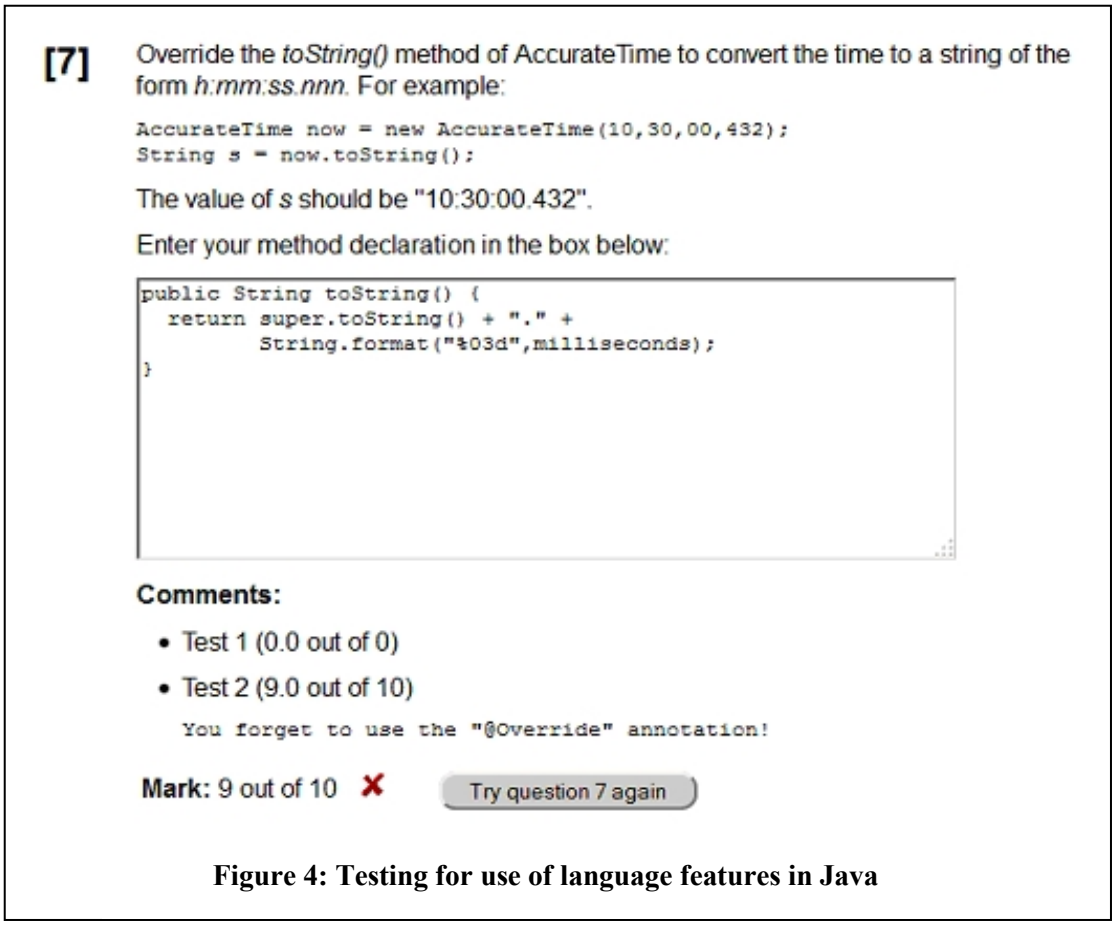


Figure 3: Feedback for a solution containing an infinite loop

To mark an exercise on a programming course, the student's submission can be embedded in a test program and compiled; the compiled code can then be executed in a sandbox (as described above) and the results compared with a set of expected results. However, there are a host of other possibilities (Ala-Mutka, 2005), including:

- Using style checkers to assess stylistic aspects of submitted programs (program length, indentation, redundant or duplicated code, and so on). This can include the use of compiler warnings to identify unused variables, implicit type conversions, and other suspect language features. Rosenthal & Suppes (2013) identified several qualitative measures for stylistic issues in program evaluation, including measuring coupling as a measure of modularity of programs and using semantic analysis of to identify meaningful variable names. Cyclomatic complexity measurement has also been used as a metric for good program design (Jackson & Usher, 1997).
- Checking for the presence or absence of various language features (e.g., by searching for the presence or absence of specific keywords), as illustrated in Figure 4.
- Measuring program efficiency by timing program execution for a variety of sets of test data. Figure 5 shows an example where efficiency is checked by measuring the number of iterations required for particular data sets compared to the number of iterations performed by model solutions with the same data sets.



[7] Override the *toString()* method of AccurateTime to convert the time to a string of the form *h:mm:ss.nnn*. For example:

```
AccurateTime now = new AccurateTime(10,30,00,432);
String s = now.toString();
```

The value of *s* should be "10:30:00.432".

Enter your method declaration in the box below:

```
public String toString() {
  return super.toString() + "." +
         String.format("%03d",milliseconds);
}
```

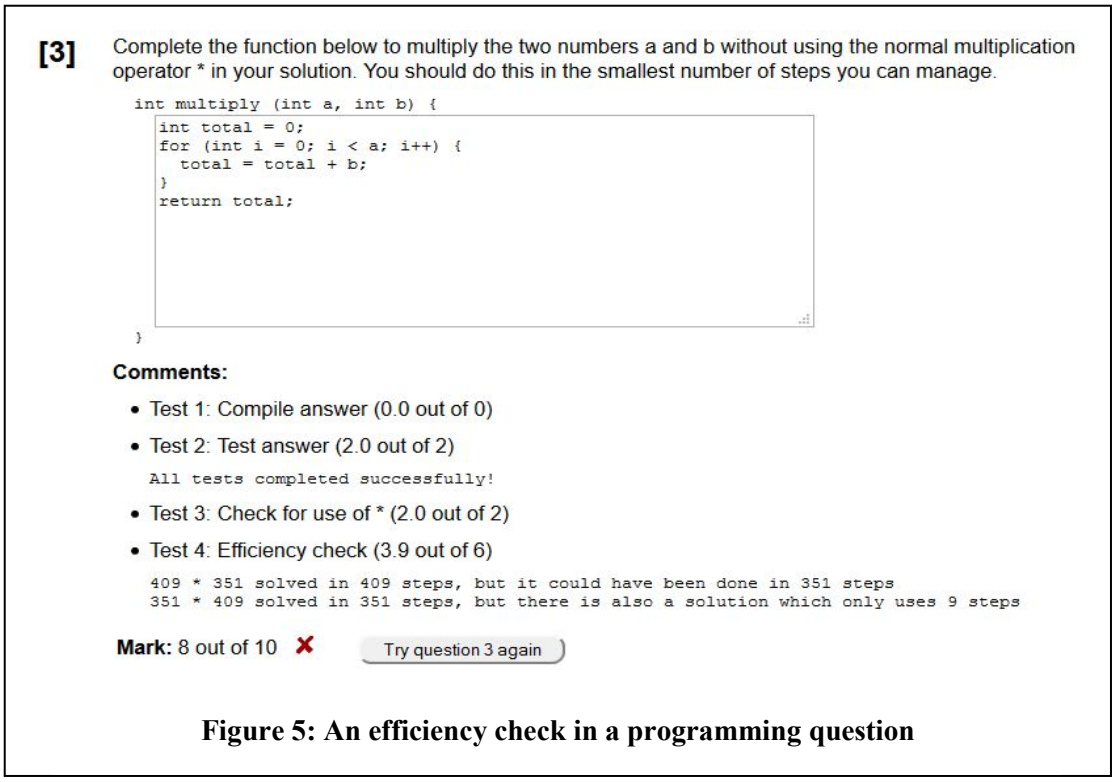
Comments:

- Test 1 (0.0 out of 0)
- Test 2 (9.0 out of 10)

```
You forget to use the "@Override" annotation!
```

Mark: 9 out of 10 ✘ Try question 7 again

Figure 4: Testing for use of language features in Java



[3] Complete the function below to multiply the two numbers a and b without using the normal multiplication operator * in your solution. You should do this in the smallest number of steps you can manage.

```
int multiply (int a, int b) {
  int total = 0;
  for (int i = 0; i < a; i++) {
    total = total + b;
  }
  return total;
}
```

Comments:

- Test 1: Compile answer (0.0 out of 0)
- Test 2: Test answer (2.0 out of 2)

```
All tests completed successfully!
```

- Test 3: Check for use of * (2.0 out of 2)
- Test 4: Efficiency check (3.9 out of 6)

```
409 * 351 solved in 409 steps, but it could have been done in 351 steps
351 * 409 solved in 351 steps, but there is also a solution which only uses 9 steps
```

Mark: 8 out of 10 ✘ Try question 3 again

Figure 5: An efficiency check in a programming question

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, & Ben-Zvi, 2013).

Result Handling

Checkpoint has mainly been used to automatically assess assignments involving free-text questions. Since assignments are marked automatically as soon as they are submitted, this makes it possible to create a 'little and often' assessment regime which would be intolerable if submissions needed to be marked manually. Each lecture can be accompanied by an exercise which addresses the topic of the lecture, although in practice we have limited ourselves to a minimum of two or three weeks between submission deadlines to avoid student exhaustion. We have also normally allowed students an unlimited number of attempts at each exercise, with the same questions being presented each time (although different students will typically have slightly different questions from each other to minimise the opportunities for plagiarism). This allows students to use the feedback produced by the system to correct any incorrect answers and resubmit them.

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. This includes tables giving a course overview, the results for an individual student and/or individual exercise. The latter lead to an analysis of which questions caused the most difficulty, as measured by the number of attempts (see Figure 6), and also to an individual attempt at an exercise; at this level a tutor can manually adjust the marks awarded if necessary.

Results found: 8 Results per page: 20 Save table as CSV Help

SORT BY:	Question ▲	Students	Average attempts	Max. attempts	Min. attempts
FILTER BY:					
1	1	18	2.9	11	1
2	2	18	1.8	4	1
3	3	18	11.0	33	2
4	4	18	4.2	16	1
5	5	17	5.1	27	1
6	6	18	5.6	46	1
7	7	17	8.9	46	1
8	8	18	6.0	21	1

Figure 6: Analysing the difficulty of questions (Question 3 was 'hard')

Other tables include information about students' last login times, and historical records of their activity, which allows tutors to monitor each student's engagement with the course as an individual. Any of these tables can be downloaded as a CSV (comma-separated value) file to allow for further analysis using a spreadsheet package or any other suitable tool.

Authoring

Although automated assessment has many benefits, it requires a far greater effort on the part of tutors to create good questions which will be marked accurately. Most of the effort required for

tutors in any automated assessment system involving free-text questions is, therefore, in the creation of questions and debugging their marking criteria. Checkpoint includes an integrated authoring system which is designed to make the development cycle as simple and efficient as possible. Tutors can either create an exercise from scratch or copy and modify an existing exercise. During this process, the author can try out the exercise as it is being developed to ensure that marking will produce the desired results and can release the assessment only when the edit/test cycle is complete. Testing needs to be conducted using at least one version of a correct solution, but it is also important to test a variety of incorrect answers to ensure that mistakes are accurately identified and that feedback in such cases is pertinent. Questions (and generic question templates) can also be stored in a question bank for later use in other exercises, categorised by topic to allow relevant items to be found more easily.

Student Feedback

In order to get the students' feedback on the courses that were taught, an anonymous questionnaire was distributed approximately two weeks before the end of the semester, after the students were nearing the end of the last assignment. The questionnaire comprised 15 Likert scale responses, with values from 1 (strongly agree) to 5 (strongly disagree), as well as some open questions inviting more general ('free-text') responses. Some of the questions are taken from earlier work (Rosenthal et al., 2013), while other questions cover points which we felt we might be able to address and improve. The open questions gave students the opportunity to raise any other issues they felt were important.

A total of 141 students filled in the forms, of whom about two-thirds of these also answered the open questions. The results show that students have a very positive view of Checkpoint.

Likert Scale Questionnaire

The students were asked if they agreed with the following statements on a scale of 1 (strongly agree) to 5 (strongly disagree):

1. Checkpoint was easy to use
2. Checkpoint was reliable
3. The marking was fair
4. The feedback was quick
5. The feedback was helpful
6. I liked being able to submit multiple attempts
7. I liked being able to work at my own pace
8. The questions were well-written
9. The questions were fair
10. The questions were hard
11. I liked being given slightly different questions to everyone else
12. I prefer having work marked by Checkpoint than by human tutors
13. Checkpoint helped me master the material
14. I liked using Checkpoint overall
15. I would recommend Checkpoint for use in other courses

The results are shown in Table 1.

Table 1: Questionnaire results

Question	Databases	C#	Java	OS	Average
1	1.39	1.39	1.53	1.26	**1.39**
2	1.48	1.57	1.75	1.42	**1.55**
3	1.22	1.65	1.44	1.16	**1.37**
4	2.22	1.90	2.58	1.71	**2.10**
5	2.30	2.33	2.33	2.52	**2.37**
6	1.00	1.18	1.03	1.03	**1.06**
7	1.04	1.20	1.17	1.06	**1.12**
8	1.65	2.10	2.39	2.65	**2.20**
9	1.13	1.73	2.03	2.03	**1.73**
10	3.22	2.67	2.47	2.19	**2.64**
11	2.30	2.37	2.50	2.26	**2.36**
12	2.22	3.00	2.67	3.23	**2.78**
13	1.61	2.61	2.14	2.35	**2.18**
14	1.35	2.00	1.63	1.52	**1.64**
15	1.43	1.61	1.81	1.39	**1.56**
Average	**1.70**	**1.95**	**1.97**	**1.85**	**1.87**
Sample size	23	51	36	31	**141**

In general, the responses fall somewhere between ‘agree’ and ‘strongly agree’. There is almost universal strong agreement with some questions (notably question 6, ‘I liked being able to submit multiple attempts’, and question 7, ‘I liked being able to work at my own pace’), whereas there is a wide variation in others (notably question 10, ‘The questions were hard’, which depends mainly on each student’s individual ability). One surprise for us was for that for question 12, ‘I prefer having work marked by Checkpoint than by human tutors’, the responses were close to neutral.

To analyse this further, we performed correlation tests between each pair of questions. The results are shown in Table 2. Unsurprisingly, there was a strong correlation (0.83) between question 6 (‘I liked being able to submit multiple attempts’) and question 7 (‘I liked being able to work at my own pace’). There was also a cluster of significant and nearly-significant correlations between questions 1 (‘Checkpoint was easy to use’), 2 (‘Checkpoint was reliable’), 13 (‘Checkpoint helped me master the material’), 14 (‘I liked using Checkpoint overall’) and 15 (‘I would recommend Checkpoint for use in other courses’). A final significant correlation of 0.51 was found between question 8 (‘The questions were well-written’) and question 9 (‘The questions were fair’), suggesting that well-written questions are perceived as being fair.

The other tests did not give any significant correlation results. It is however interesting to note the generally negative correlation between question 10 (‘The questions were hard’) and the other questions, even though these are well below the level of statistical significance. This finding might suggest that ones who thought that the questions were hard were mildly antipathetic towards Checkpoint in general.

Table 2: Correlation results

	Q1	Q2	Q3	Q4	Q5	Q6	Q7	Q8	Q9	Q10	Q11	Q12	Q13	Q14	Q15
Q1		0.44	0.12	0.06	0.10	0.09	0.11	0.16	0.29	-0.16	0.22	0.17	0.29	0.49	0.28
Q2	0.44		0.19	0.28	0.23	0.14	0.22	0.14	0.25	-0.07	0.28	0.24	0.29	0.47	0.33
Q3	0.12	0.19		0.31	0.24	0.39	0.40	-0.05	0.07	0.11	0.11	0.12	0.08	0.25	0.12
Q4	0.06	0.28	0.31		0.35	0.05	0.14	0.07	0.09	0.04	0.14	0.03	0.04	0.20	0.25
Q5	0.10	0.23	0.24	0.35		0.11	0.11	0.12	0.13	0.02	0.32	0.29	0.16	0.24	0.15
Q6	0.09	0.14	0.39	0.05	0.11		0.83	-0.06	0.11	0.01	0.05	0.07	0.02	0.12	-0.08
Q7	0.11	0.22	0.40	0.14	0.11	0.83		0.05	0.22	-0.03	0.07	0.08	0.09	0.14	0.09
Q8	0.16	0.14	-0.05	0.07	0.12	-0.06	0.05		0.51	-0.19	0.17	0.22	0.23	0.22	0.28
Q9	0.29	0.25	0.07	0.09	0.13	0.11	0.22	0.51		-0.29	0.30	0.18	0.40	0.33	0.37
Q10	-0.16	-0.07	0.11	0.04	0.02	0.01	-0.03	-0.19	-0.29		-0.07	-0.19	-0.18	-0.11	-0.05
Q11	0.22	0.28	0.11	0.14	0.32	0.05	0.07	0.17	0.30	-0.07		0.22	0.26	0.25	0.16
Q12	0.17	0.24	0.12	0.03	0.29	0.07	0.08	0.22	0.18	-0.19	0.22		0.44	0.31	0.24
Q13	0.29	0.29	0.08	0.04	0.16	0.02	0.09	0.23	0.40	-0.18	0.26	0.44		0.64	0.46
Q14	0.49	0.47	0.25	0.20	0.24	0.12	0.14	0.22	0.33	-0.11	0.25	0.31	0.64		0.58
Q15	0.28	0.33	0.12	0.25	0.15	-0.08	0.09	0.28	0.37	-0.05	0.16	0.24	0.46	0.58	

In particular, in the case of question 12 ('I prefer having work marked by Checkpoint than by human tutors') we did not find any significant correlation with other questions. The highest correlation was with question 13 ('Checkpoint helped me master the material') but the actual figure was only 0.44. We find it surprising that students are so ambivalent about having work marked by Checkpoint than by human tutors. Interestingly, our findings were different in earlier research (English & Rosenthal, 2009). When Checkpoint was introduced as an assessment in programming courses after students had already experienced human tutors marking their programs, they clearly preferred automated assessment over human evaluation. We hypothesize that this was due large class sizes where there are not enough staff involved in the marking process, which results in a much longer lag between submission of work and its return, and very often with less attention to detail in the marking and less detailed feedback. Also, when students were provided with a mixture of automated and manual feedback in a blind test, the students preferred the automated feedback (Rosenthal, 2004). Human marking is often inconsistent, and the evaluation criteria are not always completely clear (Whittington & Hunt, 1999). However, it is possible that students in this survey were not in favour of automated marking because it is often stricter than manual marking, and they felt they might get better marks from a less strict human tutor; tests of boundary cases in a programming assignment might be overlooked by a human grader, whereas they will always be detected by an automated system. More research will be needed to investigate this puzzle.

Student Responses to Open Questions

The students were also asked for other comments in five separate open questions:

1. How does Checkpoint compare to assessment on other courses?
2. What were the things you liked most about Checkpoint?
3. What were the things you liked least about Checkpoint?
4. What improvements would you recommend for Checkpoint?
5. Any other comments

Question 4 provided us with some ideas which enabled us to add some useful new features to Checkpoint, while the last question tended to reiterate the students' positive reaction to the use of Checkpoint. Here is a selection of typical responses to the first three questions:

How does Checkpoint compare to assessment on other courses?

- An efficient system as a whole. Improves my abilities. Gives confidence in learning.
- Superb.
- A superb evaluation method.
- Very good.
- Practical, efficient, fun.
- Helps practising. Encourages me to practice.
- Very helpful. Gives the student the ability to control his learning pace and gives the teacher the ability to get all the data and analyze it thoroughly. A great tool.
- Much more convenient. We can work in our own personal time and not only in the lesson. We get directed comments and it's convenient to practise the material like that.

The responses to this question were universally positive.

What are the things you liked in Checkpoint?

- That there are many attempts, and you can do it at home.
- Accessible, allows an infinite number of attempts. Questions vary and teach a lot. Gives confidence to the student.
- The feedback. Accessibility.
- That I can change the answer more than once.
- Helps practising. Motivates you to practise.
- The freedom to make mistakes and learn from them.
- That you can get a quick response and that you can do the exercise again.
- Accessible everywhere. Convenient and easy. Directs you towards the solution.
- Excellent and comfortable to work with.
- Gives the ability to work alone.
- The possibility to continue and practise until you get the maximum mark. Response times are very reasonable.
- Relatively fast. You can correct questions and edit them.
- You can trace the solution and understand the mark.
- Shows me what was not good in my answer & shows examples of what the result from the program should be.
- When failing it shows what you failed on or it gives examples what the output should be compared to the output that came out. Because of this it's very convenient to find where I made the mistake and see the cases I did not take into account.

There were many positive responses which 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 made comments similar to the last response quoted above: that when an attempt failed, Checkpoint provided feedback which helped to identify the problem and allowed them to resubmit. It is something that is very hard to achieve if marking is done manually. Checkpoint is testing each answer against a comprehensive battery of tests cases; a human marker would find it very difficult to be as thorough as this for even a single submission.

What are the things you did not like in Checkpoint?

- It was slow to process our answers
- That you have to reload all the exercise instead of just reloading the question
- The UI can be nicer/more modern

- That you have to have a fixed output to your answer
- Sometimes the system is slow
- Because we worked with Checkpoint we practiced less with BlueJ
- With complex questions it takes more time
- Sometimes the answer works in BlueJ but not in Checkpoint

Some of the responses here relate to the speed of the system; this was particularly noticeable in the database course for an assignment which involved creating tables and views, and inserting, updating and deleting records. The server system we are using is not a particularly powerful machine, and the assignment in question involves creating a separate copy of the database for each question submitted by each student. However, speed issues were observed at other times too but without any obvious reason. A more powerful machine and more server instrumentation to identify bottlenecks and hotspots would seem to be the best solution here.

Some of the responses (e.g., 'That you have to have a fixed output to your answer' and 'Sometimes the answer works in BlueJ but not in Checkpoint') arise largely because the students have done inadequate testing or have failed to do what the questions ask; for example, in the Java course, output might have been written to the console rather than being returned as the result of a method, which can give an incorrect solution the appearance of a working solution when testing it using BlueJ.

One of the dangers with automated assessment is shown in the answer 'Because we worked with Checkpoint we practised less with BlueJ'. Automated assessment is not a substitute for using other tools as learning aids, but it is where students get their grades from, so they tend to focus on it to the exclusion of other tools. Since we came across this comment, we have put much more effort into ensuring that students learn how to use tools like debuggers to locate problems, rather than just banging away blindly at Checkpoint.

Of the other comments, we note that the user interface for Checkpoint has not changed significantly for a few years, and so it could no doubt be made nicer and more modern! We hope that one of the students will one day sketch out a more precise description of what they mean by 'nicer and more modern'. In the meantime, there is more-or-less universal agreement in question 1 of the survey that 'Checkpoint was easy to use', so this is not a priority for us.

As a whole student responses were very positive, with relatively few responses to the question 'What are the things you did not like in Checkpoint?' compared to the other open questions, and many students mentioning that Checkpoint gave them confidence and motivated them to practise.

Conclusions

Automated assessment systems are being used more and more widely as the use of technology in education increases (Alber & Debiasi, 2013). Students tend to find this a natural development:

> 'It can be demonstrated that there are significant numbers of students who are now accustomed to using technology in all aspects of their everyday lives, including their studies. This also extends to using computers in their assessed work ... and are so used to using computers in their student lives that they even expect online assessment to be used.' (Dalmo & Eyre, 2008)

In the field of computer science automated assessment has been used in programming courses (Foxley et al., 1999), database courses (Prior & Lister, 2004), introductory courses on Internet skills (English & Siviter, 2000), algorithms and data structures courses (Laakso et al., 2004) and system administration courses (Baumstark & Rudolph, 2013). It has been used for grading a variety of different types of material, including sequence diagrams (Thomas, Smith, & Waugh,

2008), spreadsheets and databases (Kovačić & Green, 2012), programs with graphical user interfaces (English, 2004) and essays (Valenti et al., 2003). It has also been used for grading formal examinations as well as coursework (English, 2002; Thomas, 2003).

Checkpoint (English, 2006) is a versatile assessment framework which allows for both automated and manual marking mechanisms; it deals with the authoring of exercises and their delivery, as well as management issues such as reporting at various different levels of detail, cross-referencing exercises with course objectives, and dealing with exceptional cases on an individual basis. External reviewers such as course managers can be given access to the system on a read-only basis to allow for scrutiny, oversight, and auditing, and students also benefit from having access to a detailed record of their own attempts and results.

From the lecturer's point of view, Checkpoint is definitely very useful. It allows tutors to see which questions the students have attempted the maximum number of times, which helps to provide guidance on what the 'hard' topics are in each course, and also helps to identify common mistakes that the students tend to make. However, by far the most beneficial aspect of Checkpoint is that students can progress according to their own pace. The lab sessions are often used as Checkpoint 'help sessions', where the students can get individual assistance from the tutor with particular problems, and they can also work together on understanding and solving their problems. The tutor is able to help students individually with whichever specific exercise they are working on.

The feedback from a question normally tells the student what the input data was, and what the expected and actual results were, but it is up to the author of the question what feedback should be given in particular cases. However, it also means that when creating new questions, authors need to put considerable effort into identifying common errors and providing helpful feedback when they are detected. The students surveyed here made frequent mention of how useful the feedback was for helping them correct their solutions. Figure 3 is a particularly good example of the sort of clear and constructive feedback that is possible. Of course, feedback is most useful in situations where it can be used to correct and resubmit the assignment.

Students have expressed satisfaction with automated assessment in the past (Rosenthal et al., 2013), feeling that the instant feedback that it provides as well as consistency of marking is an advantage over manual marking. The results presented here 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 said that Checkpoint gave them confidence in learning and the motivation to practise. One student quoted earlier said, "When failing it shows what you failed on or it gives examples what the output should be compared to the output that came out. Because of this it's very convenient to find where I made the mistake and see the cases I did not take into account," and similar sentiments were expressed by many others. This feedback and resubmission cycle is something that would be very hard to achieve in a manual marking system. It also illustrates how learning best takes place: by learning from your own mistakes, and being given the opportunity to 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. Plagiarism is less of a problem as students 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. The students are in effect engaging in a battle of wits against Checkpoint, and in this situation copying someone else's answer would be an admission of defeat.

One major obstacle to adopting automated assessment is the difficulty of creating good questions which have been thoroughly tested and which provide relevant feedback. It is certainly true that in some cases it appears easier to write questions and mark them manually, as the question phrasing can be less precise and the marking can be done more impressionistically, in much the same way as in some cases it is easier to do a job yourself than it is to write and debug a program to do it. However, in the long run the reverse is true. The effort involved in marking assignments time after time, the delay in providing students with feedback, and the difficulty of supporting multiple attempts make automated assessment more attractive when it is done year after year after year. The initial investment in question development pays off quite quickly, especially when you can accumulate good questions in a question bank and either re-use them directly or use them as a basis for creating new questions on similar topics.

The other major obstacle to adopting automated assessment is a lack of belief in its efficacy on the part of teaching staff. Many people still believe that 'automated assessment' and 'multiple choice tests' are effectively synonymous (Carter et al., 2003), and even when shown how free-text questions can be marked, refuse to believe that they can assess the ineffable qualities they look for when marking by hand, in the same way as many people used to believe that computers could never play chess as well as a human being because of a lack of strategic vision and human intuition. In the light of the experiences we have described here and the extremely positive feedback we have received, we hope that this paper will help to convince others to try to use automated assessment in their own courses.

References

Ala-Mutka, K. (2005). A survey of automated assessment approaches for programming assignments. *Computer Science Education, 15*(2)*,* 83–102.

Alber, S., & Debiasi, L. (2013). *Automated assessment in massive open online courses*. Seminar aus Informatik, University of Salzberg, 2013. Retrieved July 5, 2015, from http://www.uni-salzburg.at/fileadmin/multimedia/SRC/docs/teaching/SS13/SaI/Paper_Alber_Debiasi.pdf

Baumstark, L., & Rudolph, E. (2013). Automated online grading for virtual machine-based systems administration courses. *Proceedings of the 44th ACM SIGCSE Technical Symposium*, 477–482.

Carter, J., English, J., Ala-Mutka, K., Dick. M., Fine, W., Fuller, U., & Sheard, J. (2003). How shall we assess this? *SIGCSE Bulletin, 35*(4), 107–123.

Daly, C. (1999). RoboProf and an introductory computer programming course. *Proceedings of Innovative Technology in Computer Science Education '99*, 155–158.

Dermo, J., & Eyre, S. (2008). Secure, reliable and effective institution-wide e-assessment: Paving the way for new technologies. *Proceeding of the 8th Annual CAA Conference,* 95–103.

English, J. (2002). Experience with a computer-assisted formal programming examination. *Proceedings of Innovative Technology in Computer Science Education 2002*, 51–54.

English, J. (2004). Automatic assessment of GUI programs using JEWL. *Proceedings of Innovative Technology in Computer Science Education 2004*, 137–141.

English, J. (2006). The Checkpoint automated assessment system. *Proceedings of E-Learn 2006*, 2780–2787.

English, J., & Rosenthal, T. (2009). Evaluating students' programs using automated assessment: A case study. *Proceedings of Innovative Technology in Computer Science Education 2009.*

English, J., & Siviter, P. (2000). Experience with an automatically assessed course. *Proceedings of Innovative Technology in Computer Science Education 2000*, 168–171.

Foxley, E., Tsintsifas, A., Higgins, C. A., & Symeonidis, P. (1999). Ceilidh, a system for the automatic evaluation of students programming work. *Proceedings of CBLISS '99.*

Harris, J. A., Adams, E. S., & Harris, N. L. (2004). Making program grading easier (but not totally automatic). *Journal of Computer Science in Colleges, 20*(1), 248–261.

Higgins, C., Symeonidis, P., & Tsintsifas, A. (2003). The CourseMarker CBA system: Improvements over Ceilidh. *Education and Information Technologies, 8*(3), 287–304.

Jackson, D., & Usher, M. (1997). Grading student programs using ASSYST. *Proceedings of the 28th ACM SIGCSE Technical Symposium*, 335 – 339.

Kovačić, Z. J., & Green, J. S. (2012). Automatic grading of spreadsheet and database skills. *Journal of Information Technology Education: Innovations in Practice, 11*, 53–70. Retrieved from http://www.jite.org/documents/Vol11/JITEv11IIPp053-070Kovacic1051.pdf

Laakso, M-J., Salakoski, T., Korhonen, A., & Malmi, L. (2004). Automatic assessment of exercises for algorithms and data structures: A case study with TRAKLA2. *Proceedings of 4th Finnish/Baltic Sea Conference on Computer Science Education*, 28–36.

Prior, J. C., & Lister, R. (2004). The backwash effect in SQL skills grading. *Proceedings of Innovative Technology in Computer Science Education 2004*, 32–36.

Rosenthal, T. (2004). *Automated evaluation methods with attention to individual differences: A study of a computer-based course in C*. Ph.D. Thesis, The Hebrew University, Jerusalem, Israel.

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 populations* (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.

Schoep, F. (2005). *Setting up a chroot jail for CVS*. Retrieved June 7, 2015, from http://www.ffnn.nl/pages/articles/linux/setting-up-a-chroot-jail-for-cvs.php

Thomas, P. (2003). The evaluation of electronic marking of examinations. *Proceedings of Innovative Technology in Computer Science Education 2003*, 50–54.

Thomas, P., Smith, N., & Waugh, K. (2008). Automatic assessment of sequence diagrams. *Proceedings of 12th International CAA Conference*, 319–332

Valenti, S., Neri, F., & Cucchiarelli, R. (2003). An overview of current research on automated essay grading. *Journal of Information Technology Education: Research, 2*(1), 319-330. Retrieved from http://www.jite.org/documents/Vol2/v2p319-330-30.pdf

Whittington, D., & Hunt, H. (1999). Approaches to the computerized assessment of free text responses. *Proceedings of 3rd Annual Computer Aided Assessment Conference* (1999). Retrieved June 7, 2015, from http://www.caaconference.com/pastConferences/1999/proceedings/whittington.pdf

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 Israel Academic College in Ramat Gan. 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 Israel Academic College in Ramat Gan, 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.