Towards electronically assisted peer assessment: a case study

Abhir Bhalerao and Ashley Ward
Department of Computer Science, University of Warwick
email: abhirr@dcs.warwick.ac.uk

One of the primary goals of formative assessment is to give informative feedback to the learner on their progress and attainment of the learning objectives. However, when the studentltutor ratios are large, effective and timely feedback is hard to achieve. Many testing systems have been developed that use multiple choice questions (MCQ), which are easy to mark automatically. MCQ tests are simple to develop and administer through Web-based technologies (browsers, Internet and Web servers). One of the principal drawbacks of current systems is that the testing format is limited to MCQ and general questions requiring free responses are not included because marking cannot be easily automated. Consequently, many learning tasks, such as the correctness and style of solutions to programming problems, cannot be assessed automatically. Our approach is a hybrid system combining MCQ testing with free response questions. Our system, OASYS, marks MCQs automatically and then controls the anonymous distribution of completed scripts amongst learners for peer assessment of free response answers. We briefly describe the design and implementation of OASYS, which is built on freely available technologies. We present and discuss findings from a case study which used OASYS for 240 students taking a programming class involving four assessed programming laboratories in groups of approximately forty students.

Introduction

Formative assessment plays an important role in teaching by motivating learners and providing feedback on the achievement of learning objectives to both students and tutors. However, formative assessment is confounded by large student/tutor ratios, which is an inevitable consequence of resource constraints in publicly funded higher education today. Indeed, for formative assessment to be effective, the feedback to the learner must be timely, specific to the individual, and discursive.

For example, returning the marked scripts of a class test late and then giving only a summative grade completely defeats these aims. However, to be able to give individualized discursive comments on a script and return it back to the learner in a reasonable length of time is only possible if the student/tutor ratio is low. Ideally, if resources are available and scripts are multiply marked, this would give students greater confidence in the validity of the grade and comments.

Formative assessment is a critical part of science-based curricula such as Engineering, Physics, Biological Sciences and Psychology where practical laboratory sessions are integral to the teaching and learning. Such sessions are essential to give students a 'hands-on' experience of otherwise theoretical concepts (experiential learning such as Kolb, 1984). Another laudable aspect of labs is that learning is naturally student-centred and promotes autonomy (Boud, 1988). Depending on the discipline, the learning objectives will range from knowing laboratory safety procedures, to learning to conduct and record experimental observations, problem-solving, or devising or applying abstract principles. Often these sessions are graded by spot-tests or marking of lab books (sometimes as an incentive for attendance), to make sure that safety procedures are known, and to provide learning and progress feedback.

In Computer Science, we are increasingly using supervised practical programming sessions rather than seminars to reinforce problem-solving. Our first-year undergraduate programmes currently have 240 students, so having, say, four assessed labs on any course (which is not untypical) results in about 1,000 scripts to be marked by four tutors. To be effective, these scripts have to be graded and commented on within the week, ideally before the next lab session. One of the objectives of the programming-based courses is to teach students how to program, and so, for the assessment to match this objective, we must assess programs (or at least snippets of programs) constructed by the learners. This further increases the marking burden since correct program solutions are seldom unique.

One means to address the resource issue caused by larger student/tutor ratios is by some degree of computer automation, the obvious being computer-based testing. Another approach is to incorporate peer assessment. We have therefore proposed a hybrid system which exploits the efficiency of electronic document-handling whilst achieving the quality of feedback that can only be given by humans. Computer-based assessment is largely centred on the presentation and automated marking of multiple choice questions (MCQs). Examples of such online assessment systems are the CASTLE system (*www.le.ac.uk/castle*), COSE (www.staffs.ac.uk/COSE) and TRIADS (*www.derby.ac.uk/assess/newdemo/mainmenu.html*), amongst others. The need to automate the marking currently limits the answers to highly structured responses, hence the use of MCQs. Extensions of the single-stem (single) answer (permutation MCQ or PMCQ) have been proposed to increase the scope and reliability of the testing (Farthing, 1998). The goal of automated marking of open-ended questions, perhaps using natural language-processing, is still some way in the future. In our software labs, with answers consisting of program solutions without a highly constrained specification, an automated process cannot even perform a simple test of correctness (Joy and Luck, 1998). To build a system that can provide meaningful feedback about the construction of the program is even harder (Beevers, Youngson, McGuire, Wild, and Fiddes, 1999; Thelwal, 1998). As Joy and Luck have shown (1999), simple measurements about the code, such as the number of comments, can be made but an automated process cannot perceive the more subtle aspects: choice of variable names; elegance of solution; why the code does not give a desired result. In short, the removal of the human element from the code assessment processes reduces the quality, and consequently, the validity of the assessment.

Although computers cannot make effective judgements of scripts they can certainly simplify the document management problem. The current Web and database architectures now offer a great deal of flexibility and portability in the development of computer-based assessment systems. Our solution to the problem of meaningfully assessing solutions to open-ended questions or answers which require problem-solving was to engage the group of learners themselves in the assessment process, namely to use peer assessment (Brown, 1998; Topping and Ehly, 1998).

Electronically mediated peer assessment

Peer assessment immediately raises the spectre of 'the blind leading the blind' – how can learners help each other when they themselves do not fully understand the material? To alleviate this concern we chose to use two aspects of our automated system: that it can easily anonymize and distribute multiple copies of scripts between learners, and that learners can be quickly graded on the response to 'closed' MCQ questions. The automated marking process works as follows. Each learner takes a test consisting of a number of MCQ and open-ended questions. After completing a test, each learner than becomes an assessor and is required to mark three scripts. The system ranks the learners into three groups by the total of their correct MCQ answers. This can be done immediately the script is complete. The distribution of scripts is controlled such that peers receive approximately one script from each of the good, intermediate and poor MCQ results. If necessary, the perceived ability of an assessor can be further augmented by the overall marks from a previous test or some other a priori information. After this peer marking is complete, each script will have been marked several times, increasing the validity of the marks. Furthermore, tutors can instantly view the variability of the marks given to each script. If the variance is high because of disagreement between the assessors, the script is highlighted for moderation by a tutor.

As well as the statistical safeguards in our system, the worries surrounding peer assessment can be countered by other educational benefits that a large-scale, learner-centred system affords. Feedback is given in triplicate and it is individualized and discursive. Asking learners to evaluate work is also justifiable: they are asked to read as well as write code, which is a vital skill since few programs are written entirely from scratch. It is often quoted that the best way to learn is to teach and here, in writing meaningful feedback, is an opportunity for the learner to take on the role of teacher (sometimes termed peer tutoring). Also, evaluation is an active process, encouraging reflection upon and discussion of one's own answers, thus fostering a deeper approach to learning (Donaldson and Topping, 1996). The experience of seeing multiple opinions on a piece of work (peer review as it were (Robinson, 1999)) further promotes what Perry termed a 'relativist' as opposed to a 'dualist' approach to learning (Perry, 1988). Finally, the system alleviates a greater part of the marking burden, thus allowing tutors to concentrate more on the teaching material and the moderation process.

Design of online assessment system (OASYS)

At the outset, we identified the following requirements for our computer-assisted system. It should:

- provide anonymity to all learners;
- be distributed and cross-platform;
- respond in real time;
- present learners with test, mark and view-results interfaces;
- provide tutors with authoring, moderation and administration interfaces.

As well as considering these requirements, we decided to use open-source (free) software in its design to overcome the need for licensing costs and to provide greater control of implementation choice. The architecture of the resulting system is illustrated in Figure 1. It is not surprising that a Web-server-based solution satisfies most of our requirements: access control and registration, distributed, cross-platform and the means to give learners and tutors appropriate interfaces by static and dynamic HTML pages.

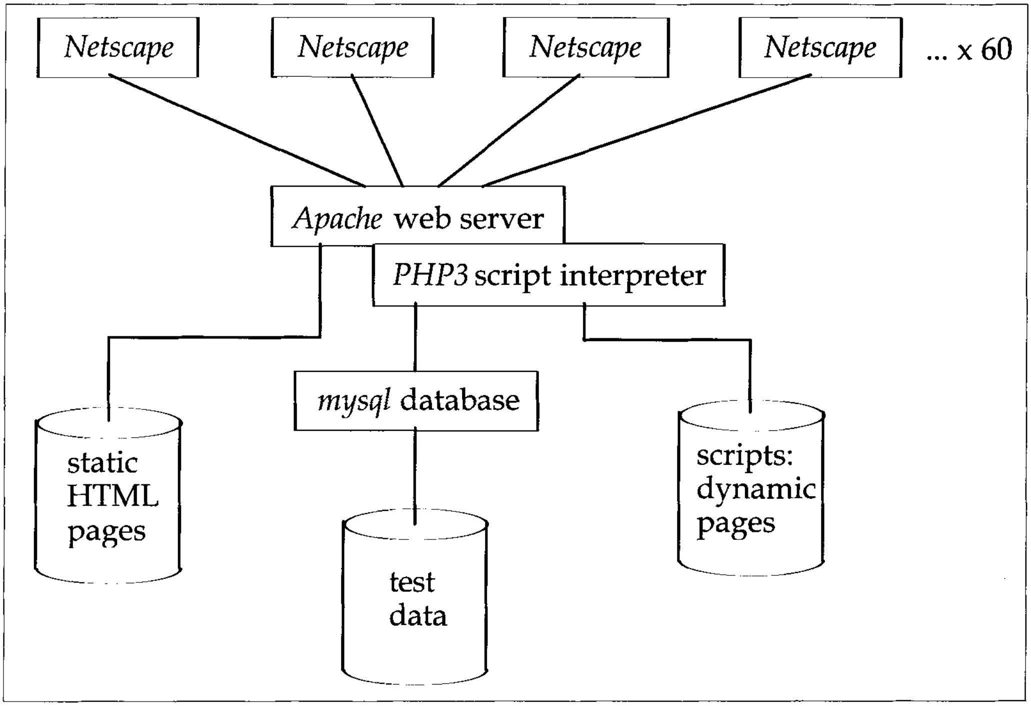
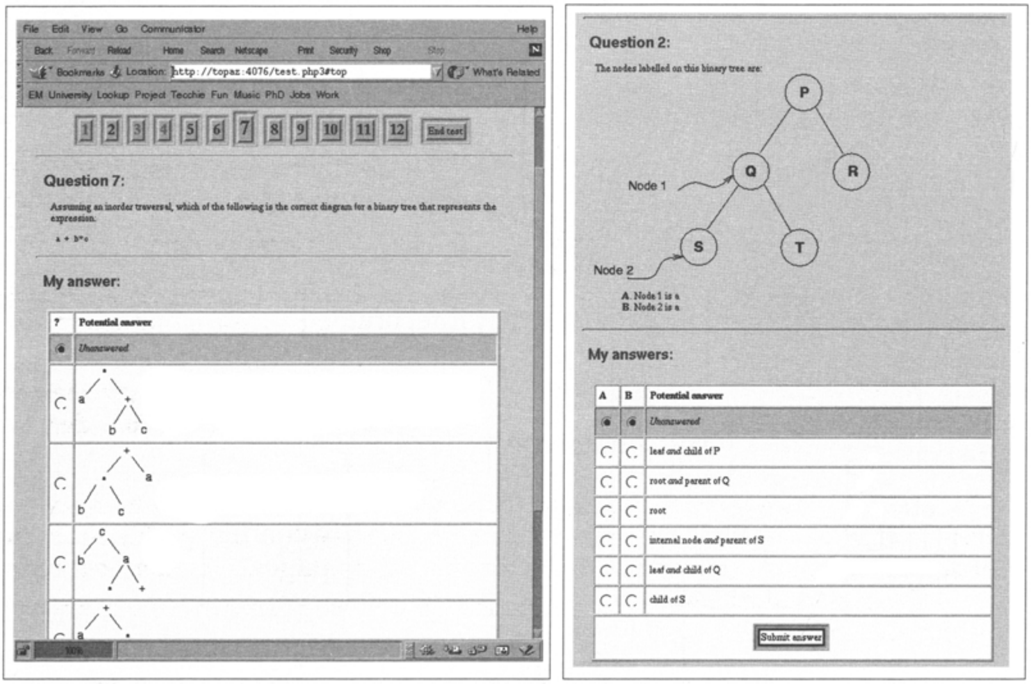


Figure 1: Technical architecture of OASYS. Standard, static Web pages are served up as well as pages generated on-the-fly by scripts written in PHP3. Test data, responses and marks are kept by a mysql *relational database. The system is built entirely from open-source software.*

A relational database, *mysql* (Yarger, Reese, and King, 1999; DuBois, 1999), was used at the backend to store the test data and record all responses. As well as providing structured storage and persistence, by means of simple queries, this enables rapid analyses of the results for script distribution, online test monitoring and moderation. The response times were minimized by using a Web-server-optimized HTML pre-processor, PHP3 (Medinets, 1999; Hilton, Willis, and Borud, 1999), as opposed to slower CGI scripting technologies. In fact, we were able to achieve near instantaneous response times for up to 100 students simultaneously being presented with test questions and the system recording their answers.

The need for anonymizing cannot be overemphasized. As well as being a statutory requirement under the regulations of the university regarding student testing, it is a requirement to store marks securely in this form under the Data Protection Act. Furthermore, part of the success of peer assessment rests on learners being free to comment on each other's work without reprisal. To use OASYS, students first register using their university ID and are given a unique password. At this point, the system creates a random internal identifier, which is subsequently used as an anonymized primary-key for all responses to and from the user.


File Edit View Go Communicator Help
Back Forward Reload Home Search Netscape Print Security Shop Stop
Bookmarks Location: http://topaz:4076/test.php3#top What's Related
EM University Lookup Project Tecchie Fun Music PhD Jobs Work
1 2 3 4 5 6 7 8 9 10 11 12 End test
P
Q
R
S
T
Node 1
Node 2
Submit answer

Question 7:

Assuming an inorder traversal, which of the following is the correct diagram for a binary tree that represents the expression:

a + b*c

My answer:

?	Potential answer
●	*Unanswered*
○	* a + b c
○	+ * a b c
○	c b a * +
○	+ a *

Question 2:

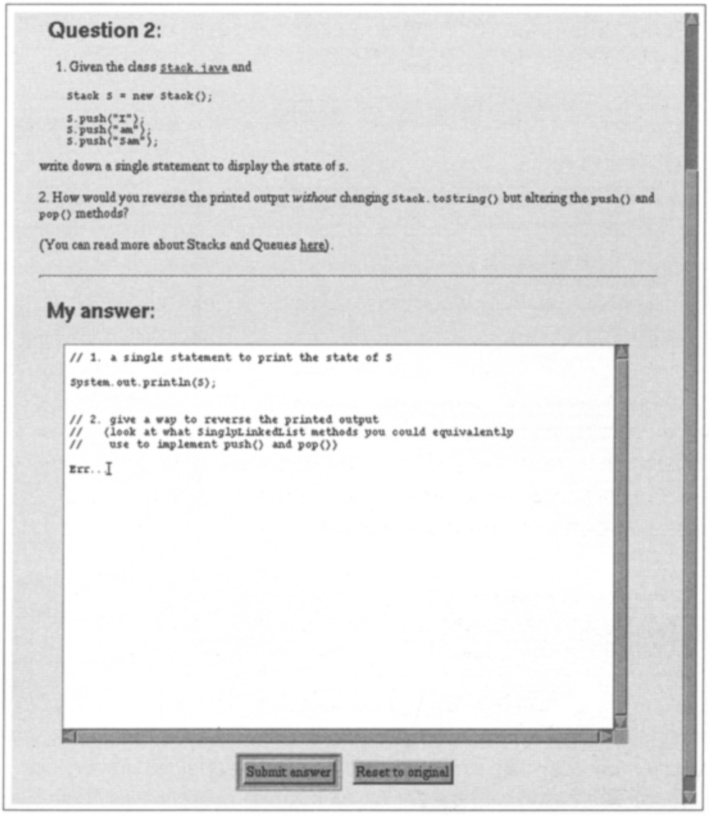
The nodes labelled on this binary tree are:

A. Node 1 is a
B. Node 2 is a

My answers:

A	B	Potential answer
●	●	*Unanswered*
○	○	leaf *and* child of P
○	○	root *and* parent of Q
○	○	root
○	○	internal node *and* parent of S
○	○	leaf *and* child of Q
○	○	child of S

Figure 2: Left – the question navigator is shown at the top. The current question has a larger button, and the colour of the question number indicates whether the question has been answered. The test need not be done in linear sequence: this learner has only answered questions 1, 3 and 4. Right - Permutational Multiple Choice Questions (PMCQs) are another question type. Potential answers to both MCQ and PMCQs are jumbled up randomly (although deterministically – this learner will always see this question this way) so that answers cannot be copied from a nearby learner.



Question 2:

1. Given the class stack.java and

```
Stack s = new Stack();

s.push("I");
s.push("am");
s.push("Sam");
```

write down a single statement to display the state of s.

2. How would you reverse the printed output *without* changing `stack.toString()` but altering the `push()` and `pop()` methods?

(You can read more about Stacks and Queues here).

My answer:

```
// 1. a single statement to print the state of s

System.out.println(s);


// 2. give a way to reverse the printed output
//   (look at what SinglyLinkedList methods you could equivalently
//    use to implement push() and pop())

Err..
```

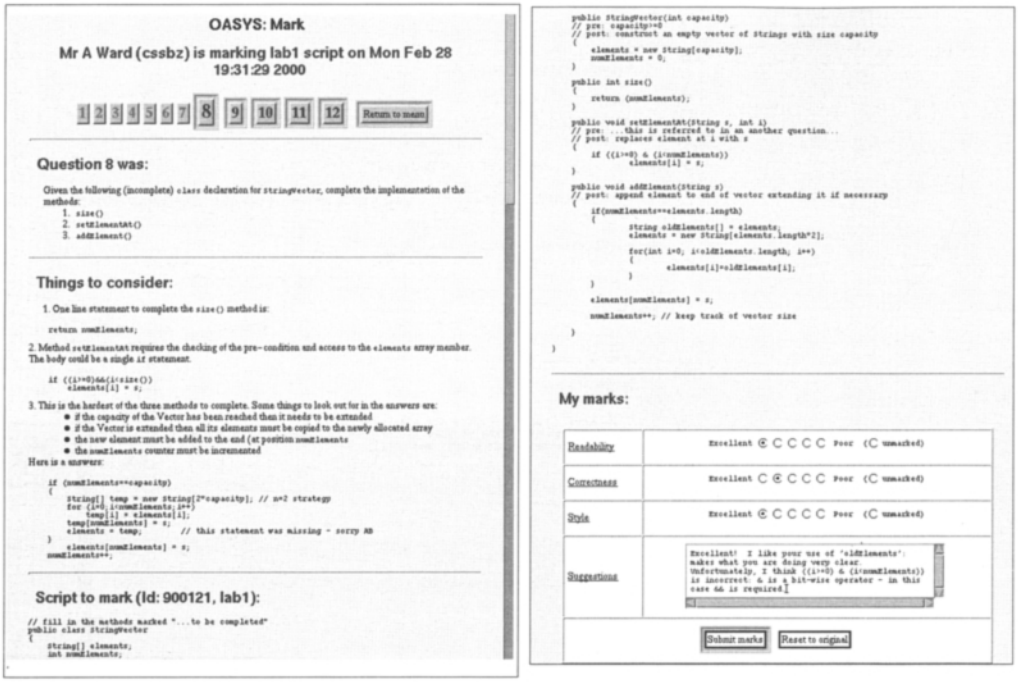
Figure 3: Entirely free answers are a major feature of OASYS – free responses are marked by other learners. Also note here the use of HTML links to supporting material, and the use of images in the question and potential answers in Figure 2.

An illustrative tour of OASYS is presented in Figures 2–7. These figures show the *testing interface* for MCQs (Figure 2) and open-ended questions during a given test (Figure 3); the associated *marking interface* when performing peer assessment (Figure 4); and finally the *see-my-marks interface*, where a learner can view comments from markers (Figure 5). Each figure caption gives more details on these interfaces and how they are used.

In designing the look-and-feel of the interfaces, we were particularly keen to mimic the paper system where possible. For example, the question navigation bar quickly allows the learner to skip between questions as questions are rarely answered in sequential order. Red- and green-coloured labels are used on these buttons to indicate whether a question has been attempted.

There is also a consistency of interface design between the testing, marking and mark-viewing interfaces. As we had several tests in the course, we employed a distinctive (but Web-safe) colour scheme for the screen backgrounds. This was useful to see which students were taking tests out of sequence because of previous absences, and so on. A distinctive end test screen was also used to show tutors those students who have signalled the completion of their test to the system.

After taking a test, students were required to mark at least three other scripts in their own time before the next lab session. During marking, the marker requests a script to be



OASYS: Mark

Mr A Ward (cssbz) is marking lab1 script on Mon Feb 28 19:31:29 2000

Question 8 was:

Given the following (incomplete) `class` declaration for `StringVector`, complete the implementation of the methods:

1. `size()`
2. `setElementAt()`
3. `addElement()`

Things to consider:

1. One line statement to complete the `size()` method is:

```
return numElements;
```

2. Method `setElementAt` requires the checking of the pre-condition and access to the `elements` array member. The body could be a single `if` statement.

```
if ((i>=0)&&(i<size())
    elements[i] = s;
```

3. This is the hardest of the three methods to complete. Some things to look out for in the answers are:
   - if the capacity of the Vector has been reached then it needs to be extended
   - if the Vector is extended then all its elements must be copied to the newly allocated array
   - the new element must be added to the end (at position `numElements`
   - the `numElements` counter must be incremented

Here is a answers:

```
if (numElements==capacity)
{
    String[] temp = new String[2*capacity]; // n*2 strategy
    for (i=0; i<numElements; i++)
        temp[i] = elements[i];
    temp[numElements] = s;
    elements = temp;        // this statement was missing - sorry AB
}
    elements[numElements] = s;
numElements++;
```

Script to mark (Id: 900121, lab1):

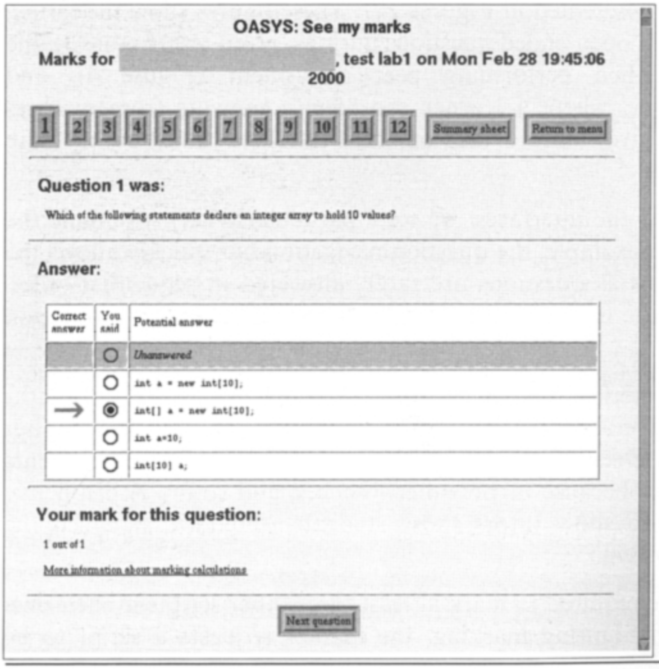
```
// fill in the methods marked "...to be completed"
public class StringVector
{
    String[] elements;
    int numElements;
```

```
    public StringVector(int capacity)
    // pre: capacity>=0
    // post: construct an empty vector of Strings with size capacity
    {
        elements = new String[capacity];
        numElements = 0;
    }

    public int size()
    {
        return (numElements);
    }

    public void setElementAt(String s, int i)
    // pre: ...this is referred to in an another question...
    // post: replaces element at i with s
    {
        if ((i>=0) & (i<numElements))
                elements[i] = s;
    }

    public void addElement(String s)
    // post: append element to end of vector extending it if necessary
    {
        if(numElements==elements.length)
        {
                String oldElements[] = elements;
                elements = new String[elements.length*2];

                for(int i=0; i<oldElements.length; i++)
                {
                        elements[i]=oldElements[i];
                }
        }

        elements[numElements] = s;

        numElements++; // keep track of vector size

    }
}
```

My marks:

Readability	Excellent ● ○ ○ ○ ○ Poor (○ unmarked)
Correctness	Excellent ○ ● ○ ○ ○ Poor (○ unmarked)
Style	Excellent ● ○ ○ ○ ○ Poor (○ unmarked)
Suggestions	Excellent! I like your use of 'oldElements': makes what you are doing very clear. Unfortunately, I think ((i>=0) & (i<numElements)) is incorrect: & is a bit-wise operator - in this case && is required.

Figure 4: Learners are required to mark three scripts in order to gain full credit. The marking page shows the original question; advice on the correct use of the mark scheme for this question; the answer to be marked (which was previously entered by another learner) and finally the marking interface. Notice that answers to the MCQs (in this test, questions 1–7) are automatically marked and so cannot be selected in the question navigator.



OASYS: See my marks

Marks for [redacted], test lab1 on Mon Feb 28 19:45:06 2000

Question 1 was:

Which of the following statements declare an integer array to hold 10 values?

Answer:

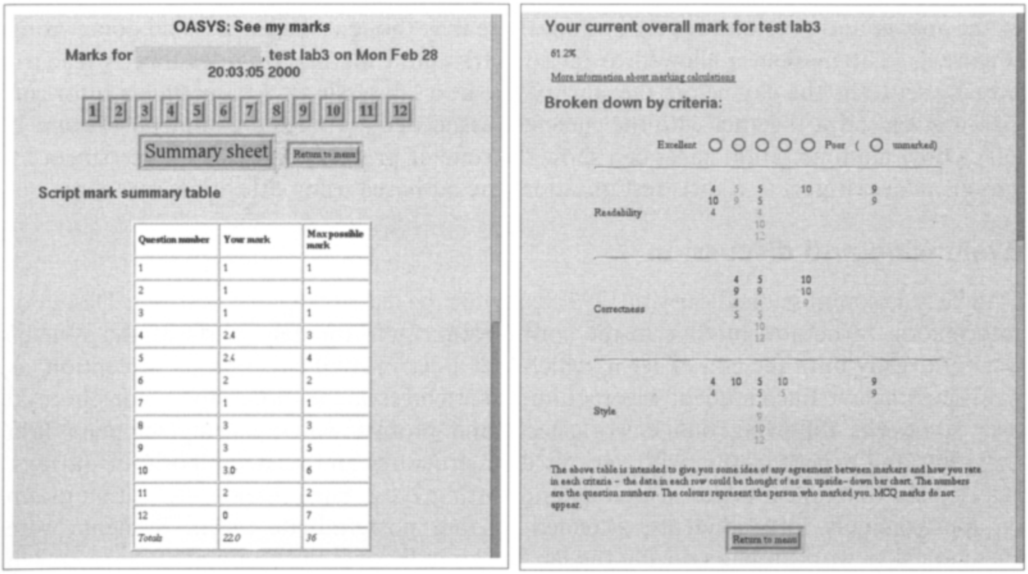
Correct answer	You said	Potential answer
	○	*Unanswered*
	○	`int a = new int[10];`
→	●	`int[] a = new int[10];`
	○	`int a=10;`
	○	`int[10] a;`

Your mark for this question:

1 out of 1

More information about marking calculations

Figure 5: 'See my marks' is where learners obtain feedback on their scripts, and can be viewed at any time from anywhere on the Internet. Marking can also be done in learners' own time. The MCQ is jumbled the same way as the learner originally saw it.



OASYS: See my marks

Marks for [redacted], test lab3 on Mon Feb 28 20:03:05 2000

Script mark summary table

Question number	Your mark	Max possible mark
1	1	1
2	1	1
3	1	1
4	2.4	3
5	2.4	4
6	2	2
7	1	1
8	3	3
9	3	5
10	3.0	6
11	2	2
12	0	7
Totals	*22.0*	*36*

Your current overall mark for test lab3

61.2%

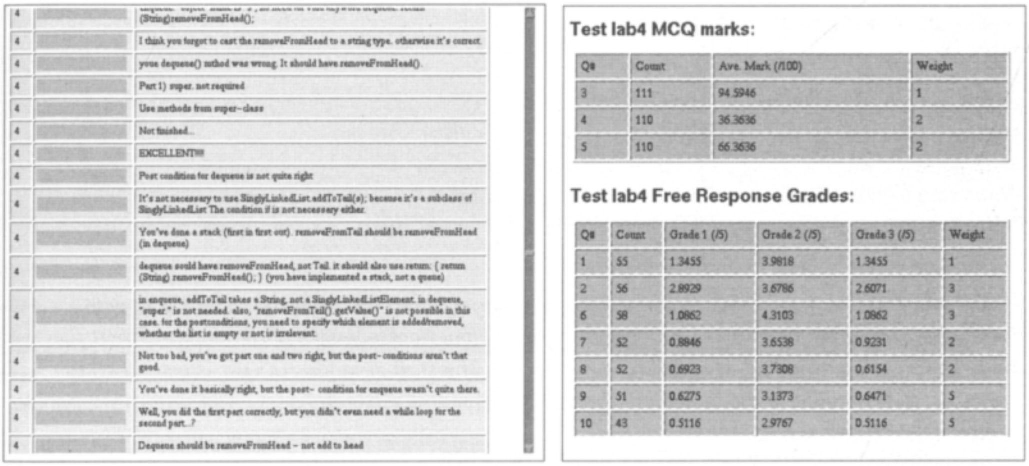
More information about marking calculations

Broken down by criteria:

	Excellent ○	○	○	○	○ Poor	(○ unmarked)
Readability	4 10 4	5 9	5 5 4 10 12		10	9 9
Correctness		4 9 4 5	5 9 4 5 10 12		10 10 9	
Style	4 4	10	5 5 4 9 10 12	10 5		9 9

The above table is intended to give you some idea of any agreement between markers and how you rate in each criteria – the data in each row could be thought of as an upside-down bar chart. The numbers are the question numbers. The colours represent the person who marked you. MCQ marks do not appear.

Figure 6: All the marking information about a script is collated for the learner in the summary sheet. This particular learner has quite a good overall result, but the overall spread of marks given by other learners (shown in the breakdown table) seems to be fairly wide, despite the lack of variance in marking by one learner who seems to have mainly marked in the middle.



4	[redacted]	[illegible] (String)removeFromHead();
4	[redacted]	I think you forgot to cast the removeFromHead to a string type. otherwise it's correct.
4	[redacted]	youe dequeue() mthod was wrong. It should have removeFromHead().
4	[redacted]	Part 1) super. not required
4	[redacted]	Use methods from super-class
4	[redacted]	Not finished...
4	[redacted]	EXCELLENT!!!
4	[redacted]	Post condition for dequeue is not quite right
4	[redacted]	It's not necessary to use SinglyLinkedList.addToTail(s); because it's a subclass of SinglyLinkedList The condition if is not necessary either.
4	[redacted]	You've done a stack (first in first out). removeFromTail should be removeFromHead (in dequeue)
4	[redacted]	dequeue sould have removeFromHead, not Tail. it should also use return: { return (String) removeFromHead(); } (you have implemented a stack, not a queue)
4	[redacted]	in enqueue, addToTail takes a String, not a SinglyLinkedListElement. in dequeue, "super." is not needed. also, "removeFromTail().getValue()" is not possible in this case. for the postconditions, you need to specify which element is added/removed, whether the list is empty or not is irrelevant.
4	[redacted]	Not too bad, you've got part one and two right, but the post-conditions aren't that good.
4	[redacted]	You've done it basically right, but the post- condition for enqueue wasn't quite there.
4	[redacted]	Well, you did the first part correctly, but you didn't even need a while loop for the second part...?
4	[redacted]	Dequeue should be removeFromHead – not add to head

Test lab4 MCQ marks:

Q#	Count	Ave. Mark (/100)	Weight
3	111	94.5946	1
4	110	36.3636	2
5	110	66.3636	2

Test lab4 Free Response Grades:

Q#	Count	Grade 1 (/5)	Grade 2 (/5)	Grade 3 (/5)	Weight
1	55	1.3455	3.9818	1.3455	1
2	56	2.8929	3.6786	2.6071	3
6	58	1.0862	4.3103	1.0862	3
7	52	0.8846	3.6538	0.9231	2
8	52	0.6923	3.7308	0.6154	2
9	51	0.6275	3.1373	0.6471	5
10	43	0.5116	2.9767	0.5116	5

Figure 7: Left: administrators can browse feedback comments entered during marking. Right: statistics can be generated in real time from the database. Question 4 (a MCQ) seems to have misled the majority of learners who have done this test so far (there are approximately another 80 learners still to complete this test at this point in time).

marked which the system then anonymizes and distributes from the 'pile' of all completed scripts within the database. Each script is presented with associated 'model' answers, hints and tips for marking and a grading scheme. Since we wanted the learners to be critical of programs we allowed them to grade each question on the readability, correctness and style of the answer and possible suggestions on where they thought the learner had gone wrong (Figure 4). The marker is allowed to revise marks until the marking deadline is reached (usually set to be the day before the start of the next lab session). At any time a tutor can view a given script together with the currently associated marks and comments (Figure 7, left). Other administration pages can show the state of progress on the peer assessment by various tables (Figure 7, right). Test questions are authored using other dynamic pages.

Evaluation and discussion

OASYS was commissioned early in 1999, ready for the teaching of the first year Design of Information Structures module in the spring term. Unfortunately, parts of the systems were not ready until the end of term, which had a detrimental effect on its perception, as we discuss below. Each student was required to attend four two-hour lab sessions in total over six weeks following online worksheets and problem-solving using template Java programs and experimenting with various data structures and related algorithms (arrays, stacks, lists, trees, hash-tables, searching and sorting, etc.). Each session was with a group of approximately forty students, attended by four postgraduate tutors. Students were encouraged to work in pairs during the lab but an individual thirty-minute test was taken at the end of the session consisting of MCQ and open-ended questions. The total credit from the lab tests amounted to 10 per cent of the module.

Academic year	Paper Tests 1998–9	OASYS 1999–2000	OASYS 2000–1
Tutors	6	6	6
Students	212	240	275
Staff time (person hours)	170	230	78
Student time (person hours)	1600	1800	2200
Average mark (stdev)	48% (2.6)	42% (0.2)	-
Best case feedback time	2 weeks	1 hour	1 hour
Worst case feedback time	4 weeks	3 weeks	1 week
Resources	2400 paper sheets	83Mb disk space	83Mb disk space

Table 1: A comparison of resources required for the paper-based testing system compared with OASYS. The staff time for 1999–2000 includes roughly four people/weeks of development and marking in addition to about seventy hours of tutoring. The staff time for 2000–1 is estimated to include a small amount of moderation and administration time (sixteen hours). The system will handle a total of 1,100 test scripts.

The first evaluation we conducted was to compare the impact that OASYS had on resource utilization with estimated figures for the paper-based assessment system we had used in the previous academic year (1998–9). The results are summarized in Table 1. The staff time for 1999–2000 includes about four people/weeks of system development in addition to about sixty hours of contact hours with students during the labs. The resource benefits of the new system will only become apparent next year (2000–1) when the staff time will require a small amount of time for script moderation and administration tasks plus the tutoring time. We estimate that students will spend about five to ten minutes marking each script (twelve to mark in total), so overall, one to two hours outside the lab sessions. Intriguingly, the average mark stayed about the same when peer marked, but the marks were less spread. The minimum feedback time is now potentially zero (in fact, it is zero for the MCQ questions which are automatically marked). The maximum feedback time was poor in 1999–2000 due to the late implementation of the feedback interface, but next year it should be at a theoretical maximum of one week (dependent simply on the timetable logistics).

At the end of the lab sessions, we used a questionnaire to gauge the perception and experience of the learners toward the online assessment system. Included in the questionnaire was a section on the students' learning style modelled on Entwistle's dimensions of Approaches to Learning (Entwistle, 1988). Some of the results of the questionnaire are given in Table 2. The amount of time students had spent on marking was not as much as we had hoped. However, 90 per cent had reconsidered their answers, which is encouraging and certainly better than never looking at the test again. The lack of timely feedback gave a predictable response. Anonymity was clearly important to these students, probably as they were all answering the same questions. The more marking they did, the better their own results became, which may in itself justify the work of peer assessment. One of the interesting results of the Entwistle Approach to Study part of the questionnaire was that 'reproducers' (who might do well on a solely MCQ-based test) did not do so well here and experienced difficulty in marking.

The final set of evaluations was conducted after the summer examinations, attempting to make statistical correlations between the students' learning approach, their lab results and their perception of the labs (from the questionnaire responses) and the overall achievement in the final examination. Where possible, these results were compared with correlation indices from the previous year. Figure 8 illustrates that the correlations between the exam (E), coursework (C) and lab marks (LM) have not changed significantly between the paper tests and OASYS, which is encouraging. However there appears to be little correlation (significantly less than 0.5) between the lab participation (LP) and the lab marks (LM) or in fact any other index. This is disappointing as we would have liked to have seen a good correlation between attainment and participation in the lab sessions reflected in both the coursework and end-of-year exams. We do not believe that this completely negates the benefits of participation in the exercise (answers on the questionnaire support this) and we would hope to improve on this aspect, perhaps by better information about the objectives of OASYS, to see a higher positive correlation in the future.

'In total, I spent this amount of time marking'	μ = 65 mins (5 mins per script), σ = 40.7
'When marking, I realised mistakes I had made in my own answers'	Yes 90%, No 10%
'I received speedy feedback on my work in the tests'	Agree 15%, Indifferent 27%, Disagree 56% (Unanswered 2%)
'Anonymous marking of the tests is important to me'	Agree 53%, Indifferent 41%, Disagree 7%

Table 2: Some findings of the post-lab session questionnaire. Learners who stated that they marked more scripts tended to receive a higher final mark themselves. Learners with a strong 'reproducing' orientation tended to receive lower marks and found marking difficult.

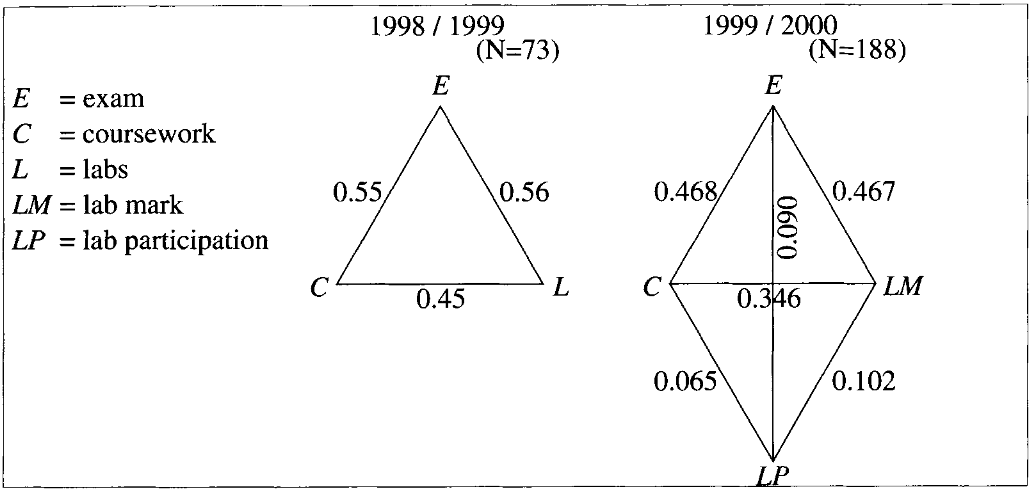


Figure 8: Correlations found in module results.

Conclusions

We have presented our experiences of implementing and using a computer-assisted peer assessment system. The aim of this study was to build a system that would give learners effective and timely feedback on the results of laboratory tests. Our system, OASYS, achieves this by the automated document management of test scripts. After taking a test, each learner is required to mark three other scripts. OASYS controls the distribution of these scripts so that they are anonymized and ranked on the results of MCQ answers. A key feature of our system is the inclusion of open-ended-questions involving problem-solving in the tests. The 'automated' marking of these questions is possible since the students mark each other's work and are given example answers and a marking scheme. If necessary, the marks and comments can be moderated by tutors before they are fed back to the learner.

Overall, the system has been successful in reducing the resource requirement for administering and marking the laboratory tests. The system has been well accepted by the students and, in general, most students have been willing participants and have felt in greater control of the testing process. Our evaluations have shown that peer assessment has been beneficial to the learners in improving their critical and analytical abilities in programming and problem-solving. The delay in completing the system last year caused a delay before students could receive their marks and comments. We do not anticipate these problems in this academic year and aim to profit further from reusing the same system.

We believe that OASYS is sufficiently flexible to be adopted in disciplines other than Computer Science. For example, simply by authoring a set of new tests, for example, for Psychology labs, the system could be reused immediately elsewhere. However, small modifications to the imposed constraints on the sequencing of the tests and marking would have to be made for non-labs specific testing. Use in a social sciences setting for peer review of complex discursive works, such as essays, would require more extensive changes to the learner interface and the a priori ranking and script distribution algorithms. None of these proposals, we think, is too far beyond the scope of our work.

References

Beevers, C. E., Youngson, M. A., McGuire, G. R., Wild, D. G., and Fiddes, D. J. (1999), 'Issues of partial credit in mathematical assessment by computer', *ALT-J*, 7 (1), 26–32.

Boud, D. (1988), *Developing Student Autonomy in Learning,* London: Kogan Page.

Brown, S. (ed.) (1998), *Peer Assessment in Practice*, Staff and Educational Development Association.

Donaldson, A. J. M. and Topping, K. J. (1996), 'Promoting peer assisted learning amongst students in higher and further education', *Staff and Educational Development Association*, SEDA Annual Conference, Paper 96.

DuBois, P. (1999), *MySQL*, CA, USA: New Riders Publishing.

Entwistle, N. J. (1988), *Styles of Learning and Teaching: An Integrated Outline of Educational Psychology*, Fulton.

Farthing, D. W. (1998), 'Best practice in non-subjective assessment', *Monitor (Journal of CTI Centre of Computing)*, 24–6.

Hilton, C., Willis, J., and Borud, B. (1999), *Building Database Applications on the Web Using PHP3*, New York: Addison-Wesley Longman Inc., USA.

Joy, M. S. and Luck, M. (1998), 'The BOSS system for online submission and assessment monitor', *Journal of the CTI Centre for Computing*, 10, 27–9.

Joy, M. S. and Luck, M. (1999), 'Plagiarism in programming assignments', *IEEE Transaction on Education*, 42 (2),129–33.

Kolb, D. A. (1984), *Experiential Learning: Experience as the Source of Learning and Development*, Englewood Cliffs, NJ/London: Prentice Hall.

Medinets, D. (1999), *PHP3: Programming Browser-Based Applications*, McGraw-Hill.

Perry, G. (1988), 'Different worlds in the same classroom', in P. Ramsden (ed.), *Improving Learning: New Perspectives*, London: Kogan Page, 145–61.

Robinson, J. (1999), 'Electronically-assisted peer review', in S. Brown, P. Race and J. Bull (eds.), *Computer-Assisted Assessment in Higher Education*, Staff and Educational Development Series, SEDA.

Thelwall, M. (1998), 'A unique style of computer-assisted assessment', *ALT-J*, 6 (2), 49–57.

Topping, K. and Ehly, S. (eds.) (1998), *Peer-assisted Learning*, Mahwah, NJL: Lawrence Erlbaum Associates.

Yarger, R. J., Reese, G., and King, T. (1999), *MySQL and mSQL*, O'Reilly and Associates.