Toward electronically-assisted peer assessment: a case study

Ashley Ward BEng MSc

Department of Computer Science,
University of Warwick

ashley@dcs.warwick.ac.uk

This document shows the slides and notes from P027, a talk given at ALT-C 2000 on Monday 11th September 2000.

Good afternoon. I'm a Graduate Teaching Assistant at the University of Warwick. I have to do a certain amount of teaching, which funds me for a PhD in Computer Science.

Internet technology provides a wonderful new communications medium,
but computers are still not teachers.

This intuitive thought led me to try an experiment at Warwick in January, which I'd like to show you. I'll take questions at the end.

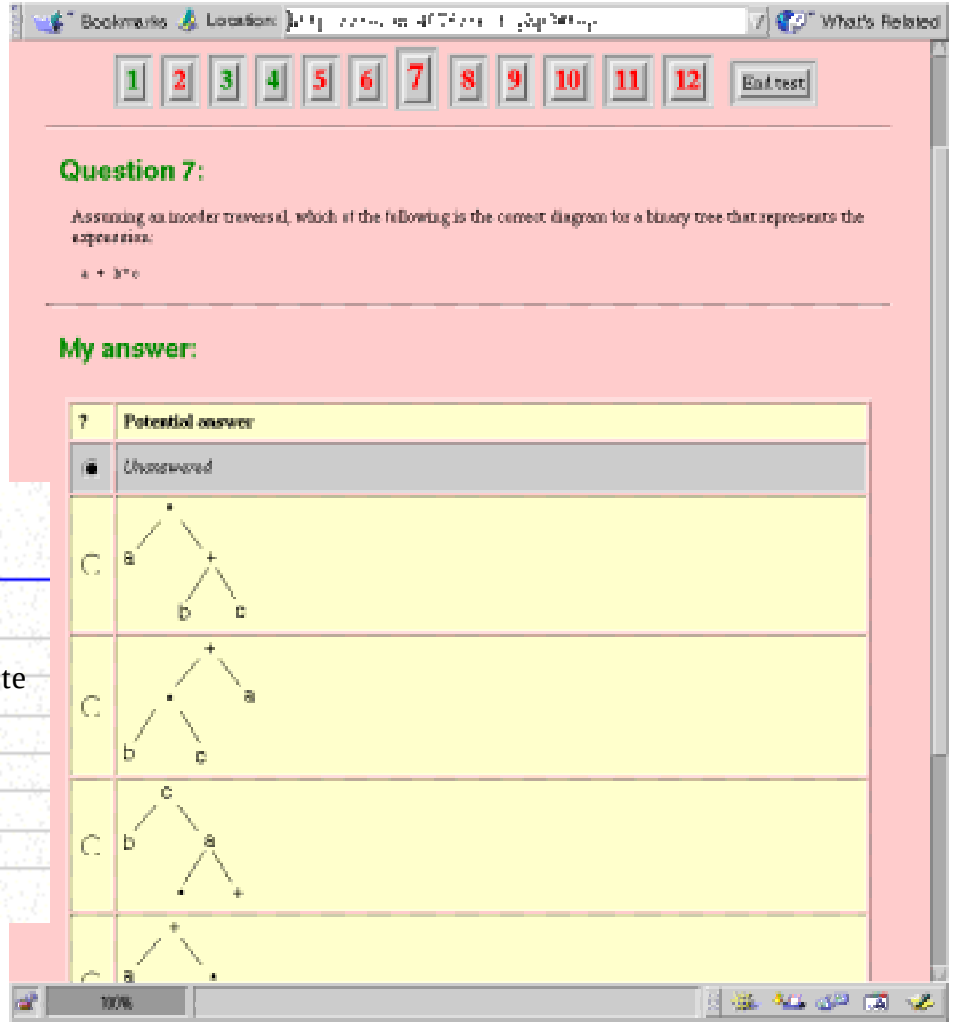
I help out with a module named *Design of Information Structures* – Java and some basic Computer Science. It's taught with lectures, an assignment, an examination and also lab sessions. More on the process later – first the system.

Interface
Process
Motivation
Analysis
Questions

This year, the first years on the module encountered this system during their lab sessions. We call it *OASYS*, short for On-line Assessment SYStem.

This screenshot shows a multiple choice question, which the student answers by selecting the appropriate radio button. They can also navigate around the test using the buttons at the top. The colours show the questions that have been answered.

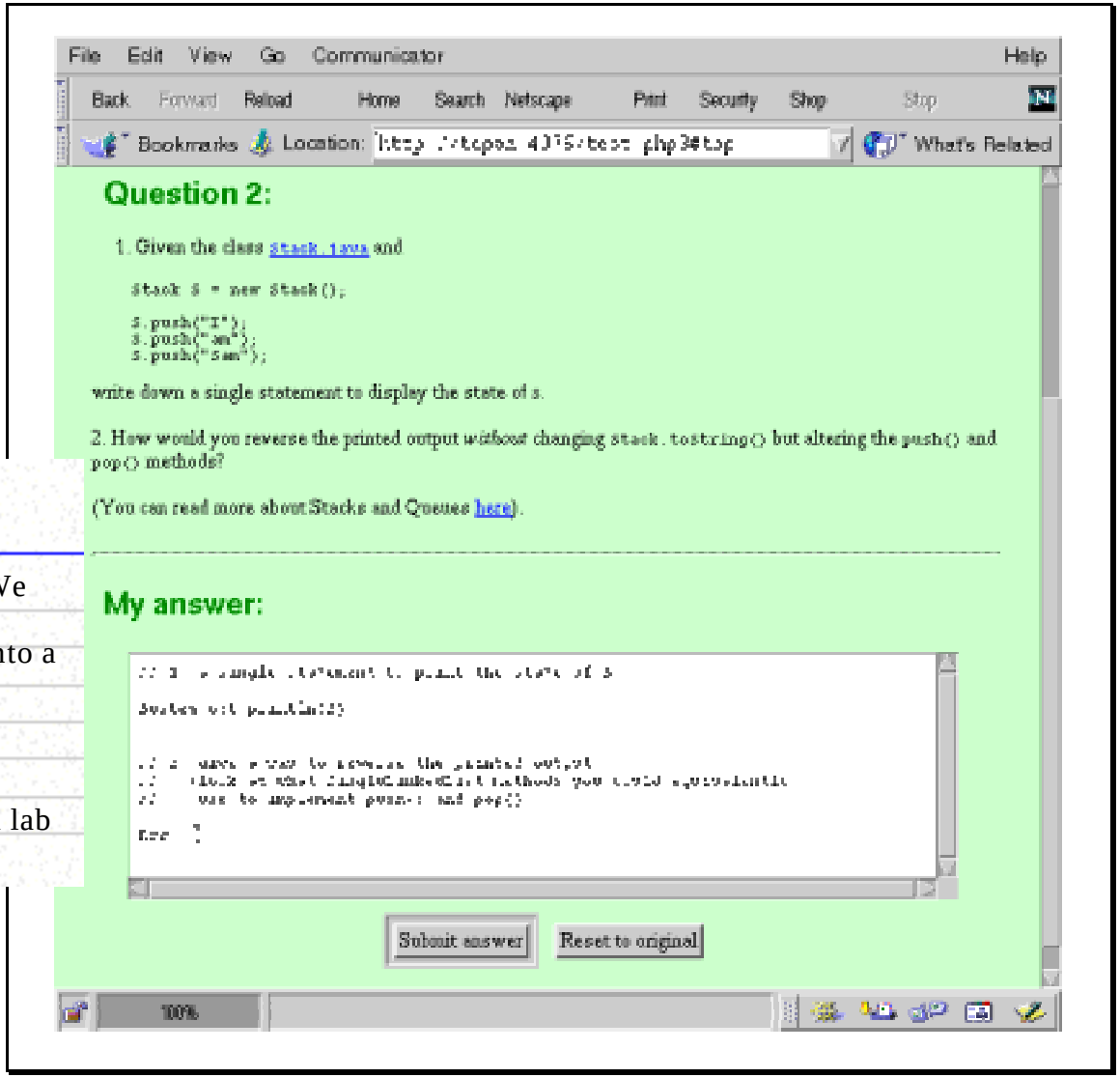
But! Multiple choice questions are not the main emphasis in OASYS.

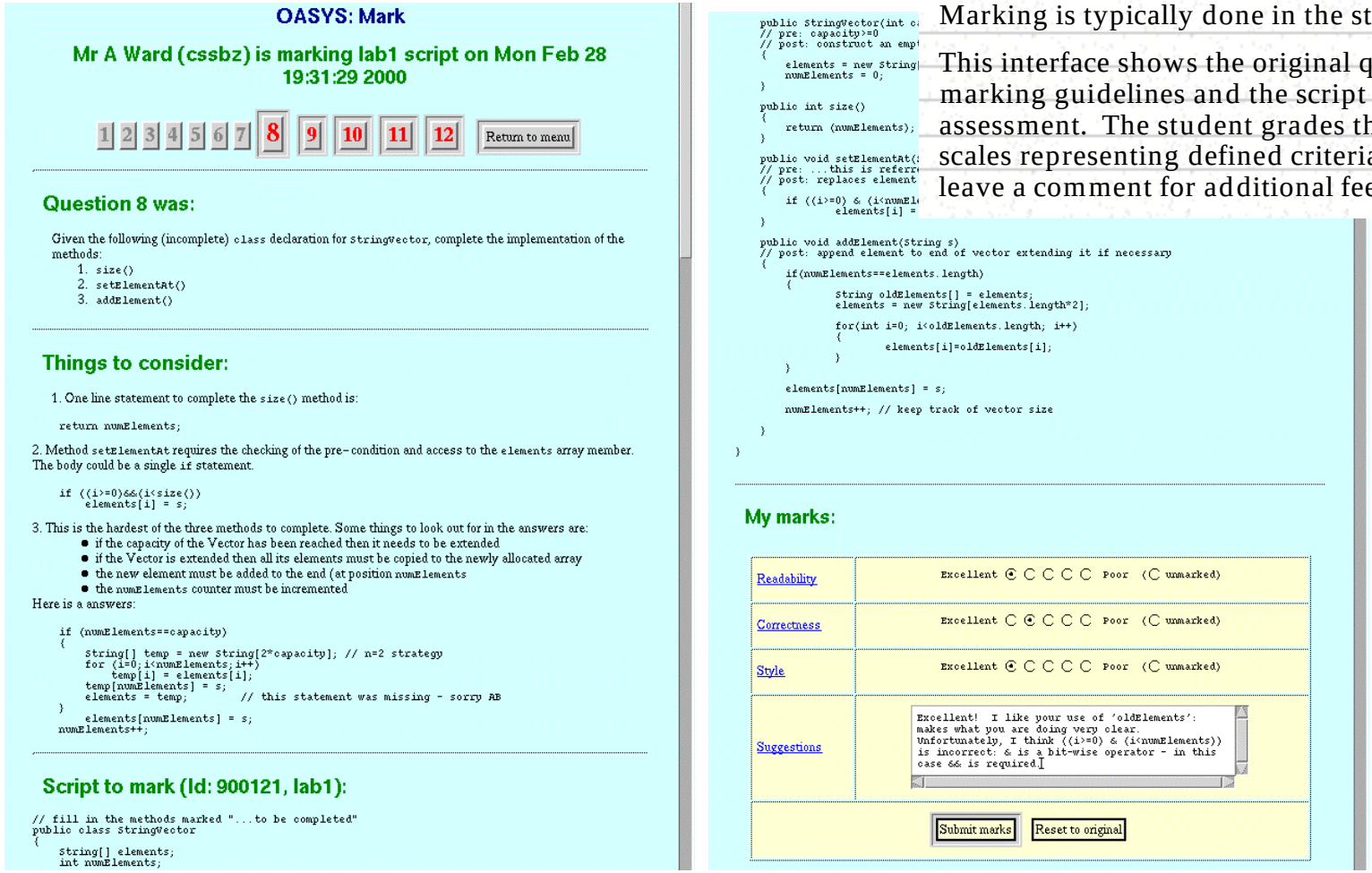


Open questions are the main emphasis in OASYS. We might ask students to outline some Java code, or explain something. Their free text response goes into a database.

Their free text response cannot be automatically marked, as it is so free-form.

The tests you are seeing happen during supervised lab sessions, 60 students at a time.





OASYS: Mark

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

1 2 3 4 5 6 7 **8** 9 10 11 12 Return to menu

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 c
    // pre: capacity>=0
    // post: construct an emp
    {
        elements = new String
        numElements = 0;
    }

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

    public void setElementAt(
    // pre: ...this is referr
    // post: replaces element
    {
        if ((i>=0) & (i<numEl
                elements[i] =
    }

    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.

Submit marks | Reset to original

Because free-form answers cannot be automatically marked, we ask students to mark one another's scripts.

Marking is typically done in the students' own time.

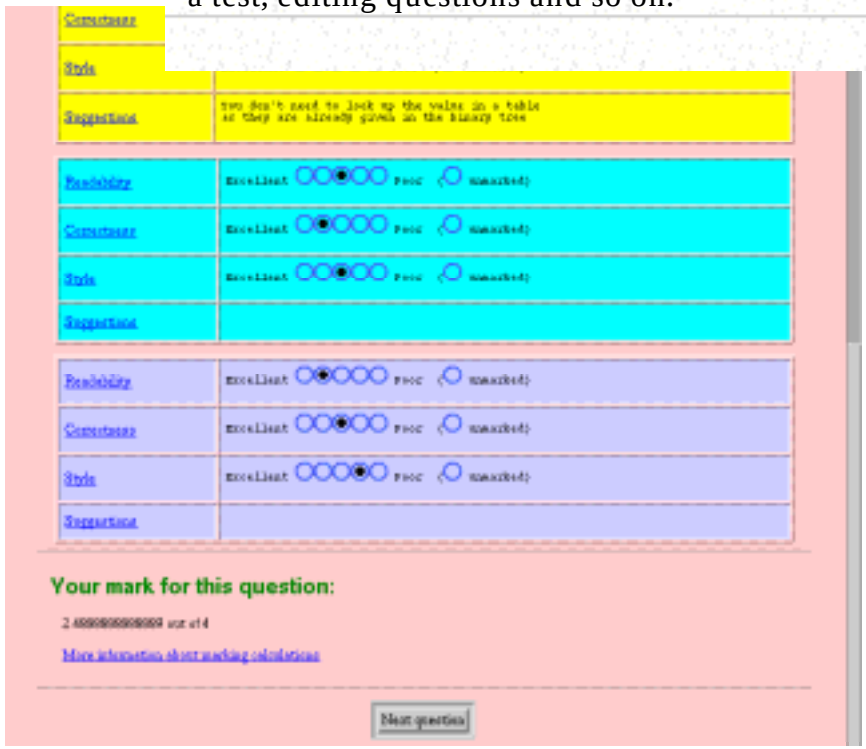
This interface shows the original question, some marking guidelines and the script requiring assessment. The student grades the answer on several scales representing defined criteria. They can also leave a comment for additional feedback.

Multiple choice questions are automatically marked, and the feedback is in the sort of interface you might expect, and is not shown here.

Once other students have marked your script, you can view the feedback in this interface.

Notice that three other students have given marks here. This is all anonymous.

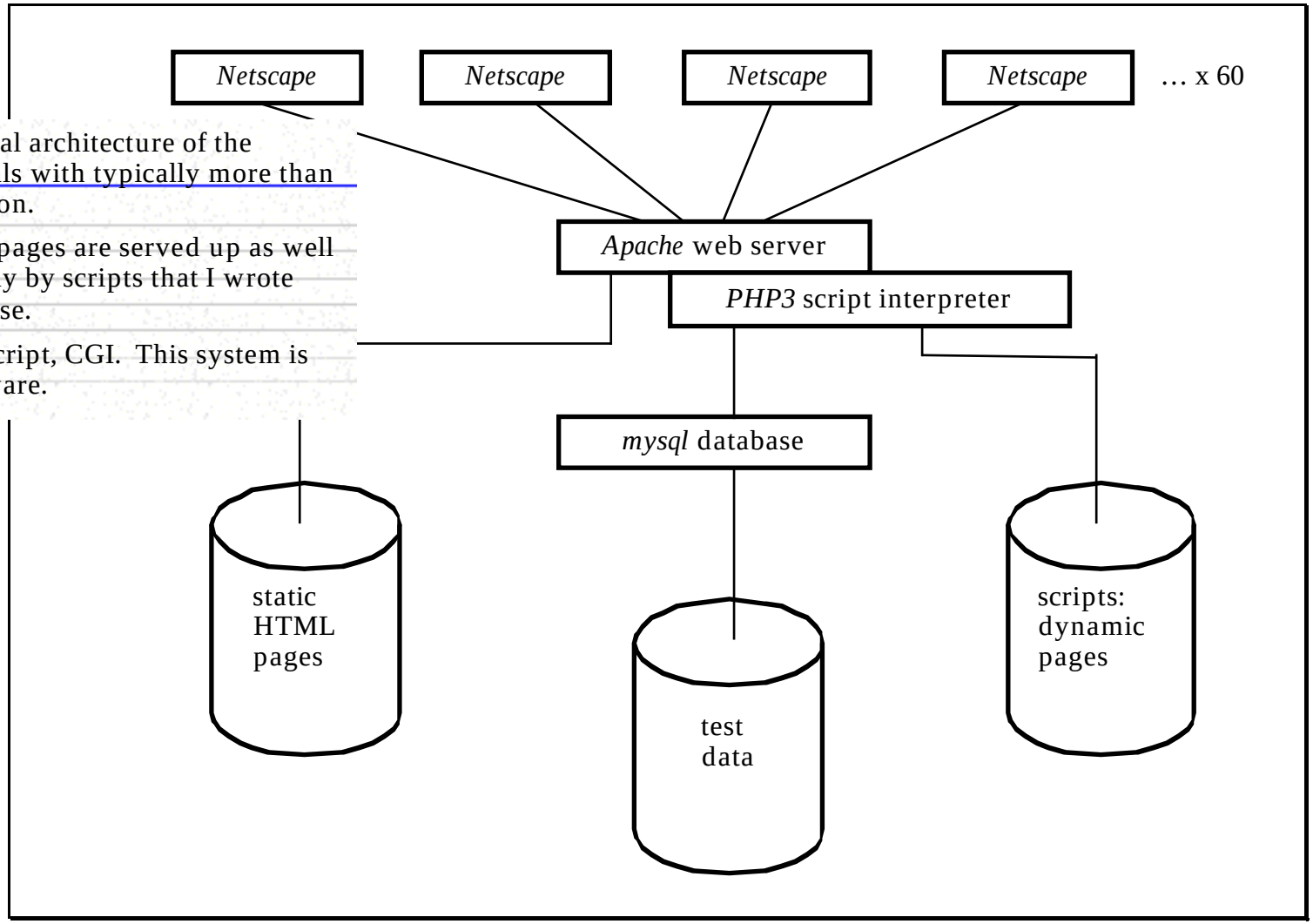
There are also other interfaces for seeing summaries of a test, editing questions and so on.



This slide shows the technical architecture of the system. The web server deals with typically more than 60 clients during a test session.

Standard, unchanging web pages are served up as well as pages generated on-the-fly by scripts that I wrote which can access the database.

PHP is similar to ASP, VB script, CGI. This system is built entirely from *free* software.



The background to OASYS goes something like this. In 1999, poorly-attended seminars were replaced with more active lab sessions, where students did a worksheet with help from demonstrators and were then tested under exam conditions. Tests were given module credit to encourage attendance. Unfortunately the 800 scripts that this generated created many difficulties when marking with only 4 demonstrators available. Eventually all we managed to give was minimal, late feedback, which was simply a number from 0-3. The tests were good for encouraging attendance, but not for giving formative feedback.

January 1999

Dear S.Tudent,

Design of Information Structures

Lectures
Assignment
Examination

4 *lab sessions*

on-line worksheet	1¼ hour	0%
paper-based test	½ hour	10% total

Copies: Approx 240 students in 6 groups

~~January 1999~~

January 2000

This year, we replaced the paper-based tests with OASYS, which I managed to put together in about 2 weeks.

We gave a small amount of credit for participating by marking scripts.

Dear S.Tudent,

Design of Information Structures

Lectures

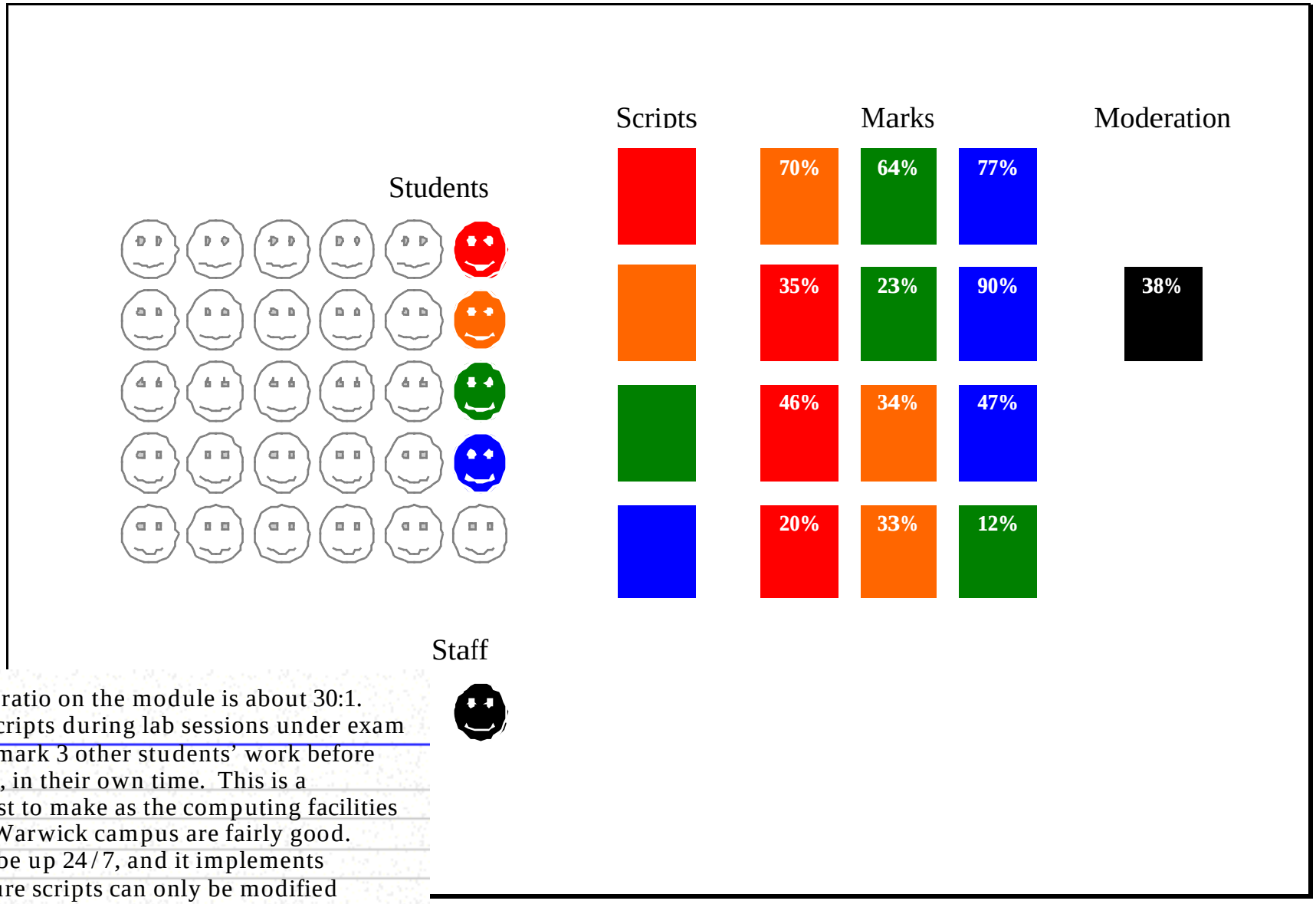
Assignment

Examination

4 *lab sessions*

on-line worksheet	1¼ hour	0%
~~paper-based test~~	~~½ hour~~	~~10% total~~
on-line test	30 mins	10% total
mark 3 scripts	1 hour?	small%

Copies: Approx 240 students in 6 groups



The staff:student ratio on the module is about 30:1. Students create scripts during lab sessions under exam conditions, then mark 3 other students' work before their next session, in their own time. This is a reasonable request to make as the computing facilities available on the Warwick campus are fairly good. OASYS needs to be up 24/7, and it implements privileges to ensure scripts can only be modified during test sessions. If the variance of marks is high, the script is highlighted for moderation. Students can also ask for moderation if they feel it is needed.

Peer assessment obviously invites criticism about “the blind leading the blind”. I attempted to justify this to the students in advance, using some of the points on this slide. OASYS is *transparent* in that it allows feedback to be seen as soon as it is given. It is important for students to *read code* as few systems in industry are written entirely from scratch. Marking is a process of active evaluation, and hopefully it encourages learners to reflect upon their own answers. By real-time analysis I mean that *scripts can be seen building up during the progress of the test*: we can see when the submission of answers start to slow down.

I made a few mistakes: we tagged the wrong MCQ stem as correct at one point. I chose not to implement a solution as it would have caused a large wart. One session was aborted by a colleague, and arranging for the students to finish up took much time. Much work is “up front”: the marking guidelines and everything *must be ready before the students touch the system*. I was late in constructing the feedback interface which caused some disappointment as we shall see. “Special cases”: I restricted the tests to the rooms set out in our timetable, which of course later changed. In general, I believe that *every restriction will meet an exception*.

There are many possible combinations of cases of unmarked, partially marked, auto-marked or unanswered questions which cause problems when collating the data into a complete script. The potential flexibility itself creates problems. Was it a summative or a formative test? We felt it was primarily formative. Unfortunately the tiny amount of credit (10% of module credit, a very small part of the overall degree) lead some learners to perceive the tests as summative.

PRO

- Fast, meaningful, human feedback
- Anonymous
- Transparent
- Read as well as write code
- Active evaluation and reflection
- Expert moderation if required
- Real-time analysis
- Can’t lose a script!

CON

- Forgotten passwords
- Mistakes are harder to fix
- Most work is “up front”
- Special cases are hard
- Scripts or questions?
- Summative or formative?

After the labs had finished, I asked the students to complete a questionnaire. The amount of time they had spent marking was not as much as I'd hoped! 90% reconsidered their answers, which is encouraging and certainly better than never looking at the test again. They didn't receive speedy feedback due to my late implementation of the feedback interface. Anonymity was important to these students, probably as they are all answering the same questions.

The more marking they did, the better their own results became: which may in itself justify the worth of peer assessment. The questionnaire I gave incorporated some of Entwhistle's Approach to Study questionnaire. Reproducers (who might do well on a solely multiple choice question based test) didn't do well here.

"In total, I spent this amount of time marking"

64.5 minutes (about 5 mins per script)

"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%

Learners who stated 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.

To compare with last year's paper-based system. We lost this year in terms of staff time, but perhaps next year we will see some benefit now that the system is built. Intriguingly, the average mark stayed the same, but the marks were less spread. The minimum feedback time is now potentially zero (in fact, it is zero for multiple choice questions which are auto-marked). The maximum feedback time was bad due to the late implementation of the feedback interface, but next year it should be at the theoretical maximum of 2 weeks, (depending on timetable logistics), but it is highly improbable that it would be that long.

	Paper	*OASYS*
Date	Jan 1999	Jan 2000
Staff	6	6
Students	212	240
Staff time	170 person hours?	230 person hours? (110 hours in 2001?)
Student time	1600 person hours	1800 person hours
Average mark (stdev)	48% (2.6)	42% (0.2)
Best case feedback time	2 weeks	1 hour?
Worst case feedback time	4 weeks, 2 days	3 weeks, 5 days (max 2 weeks in 2001?)
Resources	2400 sheets of paper?	83Mb of server space

So, to sum up, the tests are there for two reasons. We can't comfortably mark them, let alone moderate them. Some of the questions that we would like to ask can't be automatically marked, or "computers are still not teachers", the point I started with. This might seem a little pessimistic, but I don't think it is – computers can be used to greatly improve communications between humans, as I hope I've shown.

- Desire credited tests to encourage active attendance and provide feedback to learners.
- Not enough staff resource to mark (and moderate?!) 800 scripts on time with meaningful feedback.
- Not possible to automatically mark some of the questions we would like to ask.

… peer assessment

A fuller “OASYS tour” than I’ve given in this presentation is available at this URL.

This paper is also a topic for discussion on the conference web board.

http://www.dcs.warwick.ac.uk/~ashley/Research/OASYS/