BUILDING AND DEPLOYING AN EXTENSIBLE CAA SYSTEM: FROM THEORY TO PRACTICE

John Woodbury
Mark Ratcliffe
Lynda Thomas

Building and Deploying an Extensible CAA System: from theory to practice

John Woodbury
Dr. Mark Ratcliffe
Dr. Lynda Thomas

Computer Science Department
University of Aberystwyth
01970 622427
jyw@aber.ac.uk
mbr@aber.ac.uk
ltt@aber.ac.uk

Abstract

Over a two-year period we have devised and deployed over 2000 objective tests both as summative and formative tests in a range of Computer Science modules as well as diagnostic tests via the Web. From modest beginnings with an introductory module on programming for first year students we extended the service to other staff and modules including a Master's level module on programming, an introductory module to the PC and a module on computer hardware.

We recorded our experience supplying support to staff and students through a variety of documents and procedures, including addressing the practical and security issues of deploying CAA via the Web. This information is available via the Web to staff and students.

Feedback was elicited from the students after each test and this information is recorded in a database available on the Web.

We have been developing an extensible and modular system called *MAPView* (Monitoring, Access and Provision) using the development method we teach our students and written in our teaching language Java. We have used our students to develop various aspects of this system as part of their academic work and we use it as a "near experience" example in our modules on software engineering, project management and object-oriented design.

Our system is centered on the lecturer's learning objectives for a course of study. Questions and tests are constructed with explicit reference to these learning objectives. Topical feedback is provided to the students via emails automatically generated from test results, using this as a tool to revisit individual student problems. We have sought to prove that our method of system development is flexible and responsive to the individual and changing requirements of teachers.

Keywords

Java, learning technology, object-oriented, objective testing, student views on CAA, Web based assessment

Introduction

This paper reports on our experiences creating and deploying objective tests via the Web and subsequent events between September 1999 and May 2001.

During the deployment of these tests and subsequent developments that followed as a result of that experience we have recorded at intervals and by different methods feedback from our students, using this to inform future developments. We did this both to ensure that our prime responsibility to our students (described by Ramsden as providing a climate in which understanding can take place (Ramsden , 2000)) would be fulfilled, and the students' best interests protected by seeking early and clear indications of the inevitable mistakes involved in innovation. This student commentary is available as raw data on our web site (Woodbury, 2001) and is analyzed there and in a subsequent section in this paper titled *Feedback: student reaction to objective testing.*

We began as relative CAA neophytes and it was our attempts to get grips with this naivety and the unique problems of assessing students using the Web that led, often serendipitously, to renewed reflection on and enthusiasm for our own teaching.

This observation along with others is in accord with the results of Stephen and Mascia's 1997 survey of the CAA usage in HE in the UK (Stephens and Mascia, 1997) which sadly we only became aware of after our first year deploying objective tests. Reading this report was therefore an exercise in recognition and affirmation of our own experience.

We record our experiences as a phenomenon and make our data available as an example "made flesh" and illustrative of the results published in Stephens and Mascia's report. We hope that it may help others embarking or engaged in a similar exercise.

Beginnings: building the question bank and introducing objective tests

Populating the question bank

During the summer of 1999 a part-time member of staff was employed to write multiple choice questions (MCQs) for a first year module in Software Engineering.

We were aware from the beginning of Bloom's Taxonomy of Educational Objectives and its hierarchical nature (Bloom, 1956): moving through knowledge, comprehension,

application, analysis, synthesis to evaluation and tried to frame our questions to match these criteria.
In addition we felt that the distractors, i.e. those plausible wrong answers to a question we offered the candidates were an opportunity to monitor the faulty cognitive models our students currently held.

Rust (Rust, 1973) estimated in his excellent guide, “Objective Testing in Education and Training”, the cost of writing, checking, moderating, pre-testing and allocating a place in a test scheme to an objective question (known in the jargon as “items”) at about £10 or calculating to take inflation into account, about £40 in 2000 (Friedman, 2001). Those employed to create item banks are often part-timers or post-grads and relatively inexpensive but the front loaded cost for deploying objective tests, the investment made in the creation and quality audit of the question bank, remains high. Of course, in contrast to conventional subjective tests, considerable cost savings are made with the automated marking of an objective test both in terms of money and frazzled nerves.

Preparing to deploy the tests

In preparation for the tests and in response to what we understood to be good practice (Carneson et al, 2001) we developed a set of support documents both for the staff and students and made these available via the Web (Woodbury, 2001). As objective testing is likely to be a new experience for first year students we provided a web site describing what objective tests were, how they were to be assessed and provided an online example test for them to try. For lecturers we provided documents describing what objective tests were, particularly their strengths and weaknesses with respect to the familiar subjective test, along with some heuristics for good item writing. In addition we produced support documents for invigilators describing objective tests, a form to act as a log for keeping a record of incidents and their suggestions and most importantly, a protocol for conducting the tests.

These documents can be downloaded from our web site (Woodbury, 2001).

Our primary concerns revolved around question bank and test security, network reliability, possible cheating and student acceptance.

Securing the question bank and tests

As we have seen an item bank is an expensive asset to develop and we did not want it compromised by becoming publicly available. Keeping the item bank in a database secure on a central filestore is a fairly straightforward task compared to keeping a test delivered on a Web browser secure and our response to this was a combination of the procedural and technical.

During tests students are not allowed access to their network filestore, the Web browser is presented without menus or menubars and we include a clear warning in the rubric

before tests indicating that we are monitoring network traffic and attempts to subvert the system will be strictly dealt with under existing College regulations (Woodbury, 2001). No other applications, for example an emailer, can be opened other than a browser.

Invigilating a computer-based test

Unlike the standard subjective test, the invigilation of which is organized and administered centrally in our University using time-honoured ritual, there exists no such equivalent protocol for invigilating objective tests delivered via the Web. Such tests have special requirements some general and some specific to delivery issues such as scheduling.

The tests are accessed and submitted across the campus network so a fundamental requirement is high confidence that the network is reliable and predictable during the period of the test. In order to do this we need to liaise with those who maintain the system and avoid such things as routine maintenance during an examination. The system support for our network is run by another organization on campus but it would be fair to say that as Computer scientists we share many cultural assumptions and speak the same language. In addition there are many personal connections between our department and the support services as they often recruit from our staff and students. This makes communication and the necessary cooperation easier. If we give sufficient notice then the support services defer any operation that may interfere with our tests and remain on standby should a problem arise. Even so there were several incidents when students were disconcerted for example, when automatic virus checkers started running.

Although our experience in this respect has been good, anecdotal, but illustrative of the kind of problem that can arise in a computer mediated environment, we offer the following incident that occurred during one of our test series.

Delivering tests in a networked environment

The summative tests are presented to all the students at the same time in several different venues whereas our formative tests are delivered in the same venue over the course of a week. During the half hour before a summative test it is prudent to scrutinize each workstation room where tests are to be deployed for such things as the number of working machines against the number of candidates scheduled to take the test in that room. During one of these checks it was discovered that a number of machines had a warning dialog couched in the impenetrable jargon and alarmist language so neatly described by Norman in The Psychology of Everyday Things (e.g. “Fatal error”, “kill process” Norman, 1988) as an occupational hazard for computer program developers. In this instance the message was a countdown message to a routine operation warning the user that they should save their work. However, because no one had been using the computers for some time, five of these messages were stacked one behind the other on the monitors of a number of computers. As only the

time part of the message was changing in each dialog, dismissing a dialog appeared to have no effect, contrary to good HCI (Human Computer Interaction) practice: an information dialog should go once acknowledged (Schneiderman, 1998). This is the kind of event that can unsettle an already nervous candidate. It certainly unsettled the invigilator.

The protocol for "Launching" a computer-based test

Another difference we discovered between computer-based tests and conventional tests is the time overhead accrued in setting up the tests on the students' computer screens. Logging in and navigating to the appropriate pages, taking the students through the password protection login protecting the test, taking them through the procedure of entering their details in the test preamble and reading the rubric can typically take 10 minutes. This may appear a long time for anxious students to wait for a test to start but we have never received any complaint about this part of our administration of the test (Woodbury, 2001). The invigilator needs to take firm control leading the students carefully through each step while remaining sympathetic and calm during the inevitable problems. We always have two invigilators for a workstation room with typically about 30 students. One invigilator does the "launch" whilst the other acts as a roving troubleshooter.

Feedback: student reaction to objective testing

One of our basic requirements was to carry the students with us and our strategy was to engage them in the setting up and refining of the tests. We made explicit appeals asking them to help us improve our provision (cf. the document "Scratch pad" Woodbury, 2001).

To this end we elicited feedback in several different ways.

Feedback using "scratch pads" on each test for a survey: 1999-2000

We provide for all tests a form called a "scratch pad" for "workings out", collecting these at the end of the test. We take the opportunity to ask the students:

1. to note down when they started the test and when they finished,
2. to note whether they used the online support documentation,
3. to note down on a scale of 1 to 5 how difficult they found the test,
4. to give us feedback about any problems they had with the questions or errors they detected,
5. to give us, towards the end of the academic year, their opinions on objective tests as a way of assessing them and how we administered them.

We have entered the results from this first year survey into a database and made this available on the Web (Woodbury, 2001). Reference numbers in the following table index individual entries that can be read in their entirety, including original spelling and grammar, in those Web pages.

Table 1: Students perceived difficulty of tests

Difficulty Level	Responses	%
Very Difficult	24	10.3
Difficult	108	46.3
Moderate	96	41.2
Easy	5	2.2
Very Easy	0	0.0
Total	233	

Table 2: Results from the “scratch pad” survey for a first year course of 125 students

Sept 1999 – May 2000			
	Responses	True	%
Used online support documentation	243	66	27
Used “scratch pad”	359	255	71
Gave feedback on questions	359	182	51
Used “scratch pad” for “workings out”	359	128	36
Positive feedback on testing		41	
Negative feedback on testing		33	
Positive and negative feedback on testing		22	

Table 3: Positive feedback on tests

		Reference numbers
Less stressful than conventional tests	6	276, 277, 283, 290, 291,324
Easier for examiner to mark	3	276, 284, 331
Provides quicker feedback	1	281
Less writing is involved	1	276
Able to demonstrate knowledge	1	282
Tests knowledge and understanding	4	283,284,294,313
Preferred to subjective examinations	17	283, 285, 287, 289, 292, 295, 297, 313, 317, 320, 327, 328, 329, 330, 331, 332, 334
Prompts continuous revision	11	280, 287, 288, 295, 298, 299, 312, 315, 318, 320, 334
Easier for non native speaker	1	299
Tests and questions concise and clear	1	300

Prefer working on a computer	1	300
Provides information of progress to student and lecturer	1	280
Easier for people with a learning disability	1	289

Table 4: Negative feedback on tests

		Reference numbers
Needed to be more difficult	2	175, 294
Poor state of equipment (e.g. dirty mice)	1	275
Questions were tricky/ambiguous/faulty/poorly written	4	8, 286, 316, 360
Not enough feedback from tests/individual questions	5	279, 285, 292, 311, 315
Difficulties with presentation (long questions required scrolling (4))	9	41, 290, 296, 314, 331, 332, 338, 340, 351
Would like both types of exam	1	312
Would prefer written exam	2	279, 319
Too constrained: not stretching enough, not enough freedom to express understanding	7	279, 296, 313, 314, 317, 318, 319
Would have liked fewer questions	2	334, 338

We used the information about when students started and finished exams to calculate how long questions took to answer on average and this allowed us to fine tune the number of questions to the length of time proposed for an exam (about 1.5 minutes/question).

We discovered that less than one third of the students had consulted the online documentation by the end of the first semester when we stopped asking the question.

A large number of students had used the “scratch pad” in some way and there may be valuable pointers in the things that students choose to “work out” towards the things they have difficulty with and how.

Just over half the students gave us feedback on the questions themselves. This information ranged from pointing out spelling errors and typos, suggestions regarding the layout and presentation of questions to suggestions that certain questions or the choices offered for the answer were in error (unintentionally). Sometimes the questions were poorly worded or the answer offered faulty and this was another valuable dimension to our quality audit. However, sometimes the errors the students were

pointing to turned out to be misconceptions on their part and this provided interesting evidence of the faulty models students had in their minds of the material we were presenting them (Laurillard, 1999).

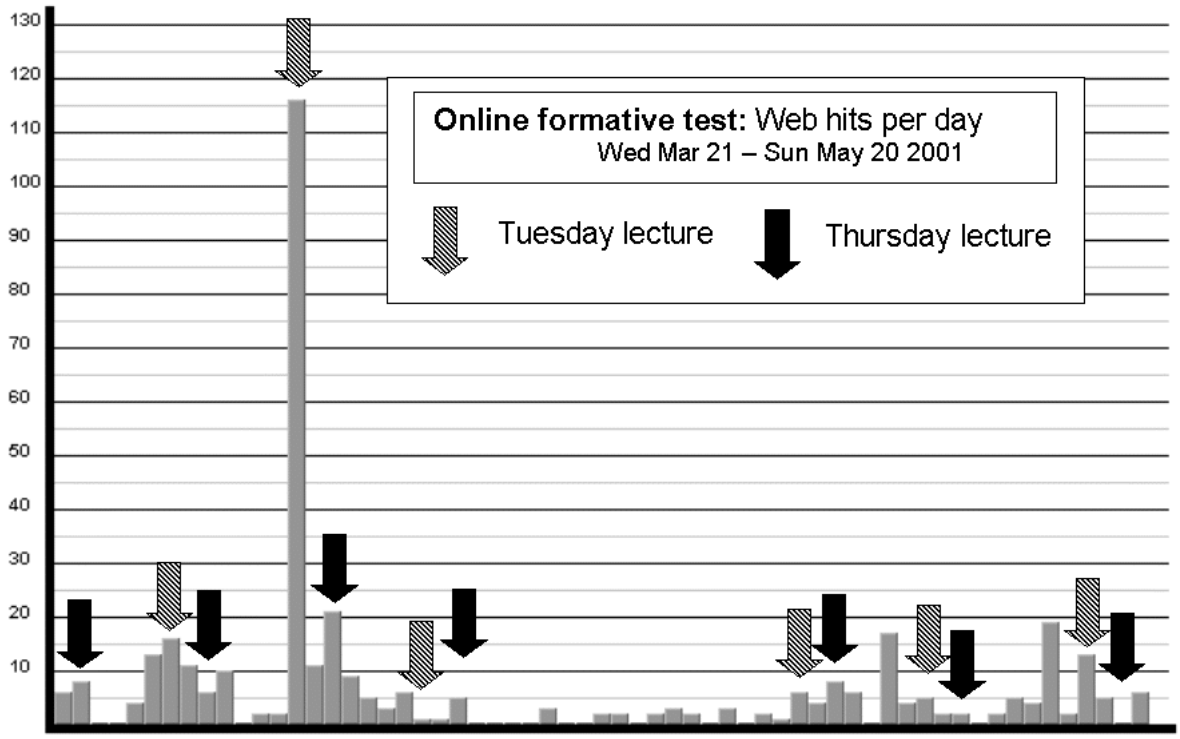
Mid-module feedback: 2001

For this first year module on Computer Architecture the students were provided with two versions of an online objective test. One version gave feedback on each question answered and the other general feedback on submission of the test. Half way through the course and mid-way through a lecture students were asked to complete two yellow Post-Its, labeling one “Good” and the other “Bad”. They entered anonymously their critique on the course and lecturer, sticking the *Post-Its* to the lecture room door on leaving. Fifteen percent cited the online tests as a useful revision aid.

Feedback from end of module student questionnaires: 2001

The figure below illustrates the correlation between the dates when the lecturer for this course reminded the students during a lecture of the online test existence and the test’s subsequent use.

Figure 1: Web usage of a formative test for a first year module on Computer Architecture: generated using MAPView



End-of-module student questionnaire: 2001

Student questionnaires are provided online at the end of modules.

This feedback comes from the end-of-module questionnaire for the first year Software Engineering module that was the subject of the "Scratch Pad" survey in the year 1999/2000 referenced above. This module is assessed 50% on coursework and 50% on the best result of two mid-term tests and has 120 students. The students were asked what they thought of the way they were assessed: of the 57 respondents to the questionnaire 12 rated the objective tests as good/excellent, 6 said they appreciated the frequent feedback on how they were doing.

Building on experience: MAP making

What we had learnt from our first year deploying our objective tests was an amalgam of our own observations and the feedback we had from our students.

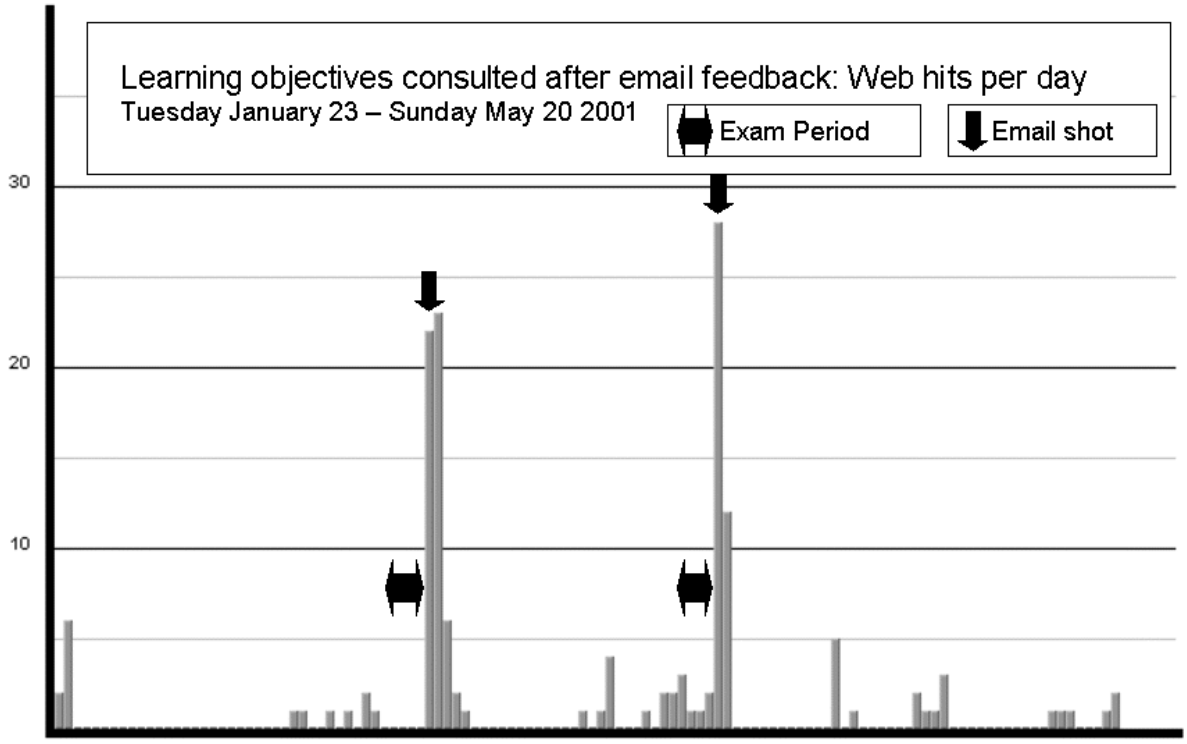
The limitations of our proprietary software leads us to experimentation

We had from the beginning met the limitations of the proprietary software we were using by creating our own solutions. With the original software the user interface for entering questions and creating tests was found to be too small as well as irritating in its unforgiving functional behavior: e.g. the questions for tests could only be listed in the order they had been entered into the underlying database, a list of questions that had been compiled for a test had to be re-entered if a question had been omitted and tests could not be saved and recalled at a later date for editing and reuse. We built our own interface and added the desired functionality to meet these problems.

Although the results from a test were automatically calculated within half an hour of the test's completion we could not inform the individual students of their results. We added software that generated personalized email shots for students telling them what their score had been and whether they had passed or failed. We suggested to those that had failed that if they were concerned they could see the lecturer concerned.

This proved to be popular with the students and we began to think of what further possibilities there might be. In the following year we provided more focused feedback and we were able to analyse a student's performance on a test and generate an email pointing to them with links to the learning objectives they needed to revisit. Figure 2 plots the reaction of the students to these emails.

Figure 2: Students accessing a module's learning objectives after email feedback from a test: generated using MAPView



We consolidated work we had already begun on making the Web presentation of the tests more user-friendly in accordance with good practice by changing the colour scheme and layout of the questions and answers, adding extra navigation so that a test could be taken without using scrollbars (Schneiderman, 1999).

Putting our experience in context

The first year cohort of students presents a broad demographic profile, bringing a wide range of past experience and skills, and a variety of learning styles.

We felt we needed to monitor students' progress more closely, particularly first year students who may bring poor study habits and a surface approach to their studies (Ramsden, 2000), in order to intervene before their academic careers were compromised. As our students work increasingly in a digital environment as described by the director of the Media Laboratory of the Massachusetts Institute of Technology (Negroponte, 1996) we looked there for indicators that would give an early warning of pending trouble, focusing on their attendance and performance in practicals.

The nature of higher education and the position of teaching have changed over its recent history.

“Most discussions, and most applications, of PIs [Performance Indicators] to higher education have been concerned with research performance. The stress on research to the exclusion of teaching (and related functions of higher education, such as the maintenance of values associated with tolerance and diversity, and service to the community) appears to be leading to the predictable consequence of a shift in effort from teaching to research..” (Ramsden, pp. 236-7)

The introduction of this accountancy approach to the assessment of teaching with its increasing emphasis on accountability and staff appraisal has led to increased administration overheads and, added to increased teaching commitments, escalating workloads.

Some basic requirements for a solution

We wanted to provide a tool that would allow all those engaged in the teaching enterprise a service equivalent to that provided by integrated office tools; the combination of word processors, databases and spreadsheets that we have become familiar with. Such a tool would have to be flexible and adaptable to the changing demands of reflective teaching and extendible in order to easily incorporate innovative ideas.

The introduction of educational technology in the UK has a chequered history littered with failed and counterproductive projects driven by technological determinism (Beynon and Mackay, 1991; Beynon and Mackay, 1993). As Computer scientists we had to be aware that we are engaged in a discipline that, although it addresses a wide spectrum of interests, has at its heart a dominant technology and that we may be particularly suseptible to the sirens of this determinism: the drive and exhilaration of developing the tool may by degrees overshadow the purpose.

Description of MAP and MAPView

Monitoring, **A**ssessment and **P**rovision were the three key tasks that we identified from our first year’s experience and alludes to the responsibility the teacher has in supplying guidance and support in the form of a cognitive map of a subject area.

MAPView is the application for implementing MAP, its interface composed of a series of tabbed folders that can be added or removed according to the user’s needs. Each tab provides a set of related services, for example, the creating, editing and publishing of objective questions and tests. These tabs are described in Table 5 and illustrated in Figure 3.

Figure 3: MAPView: Question bank and test management tab.

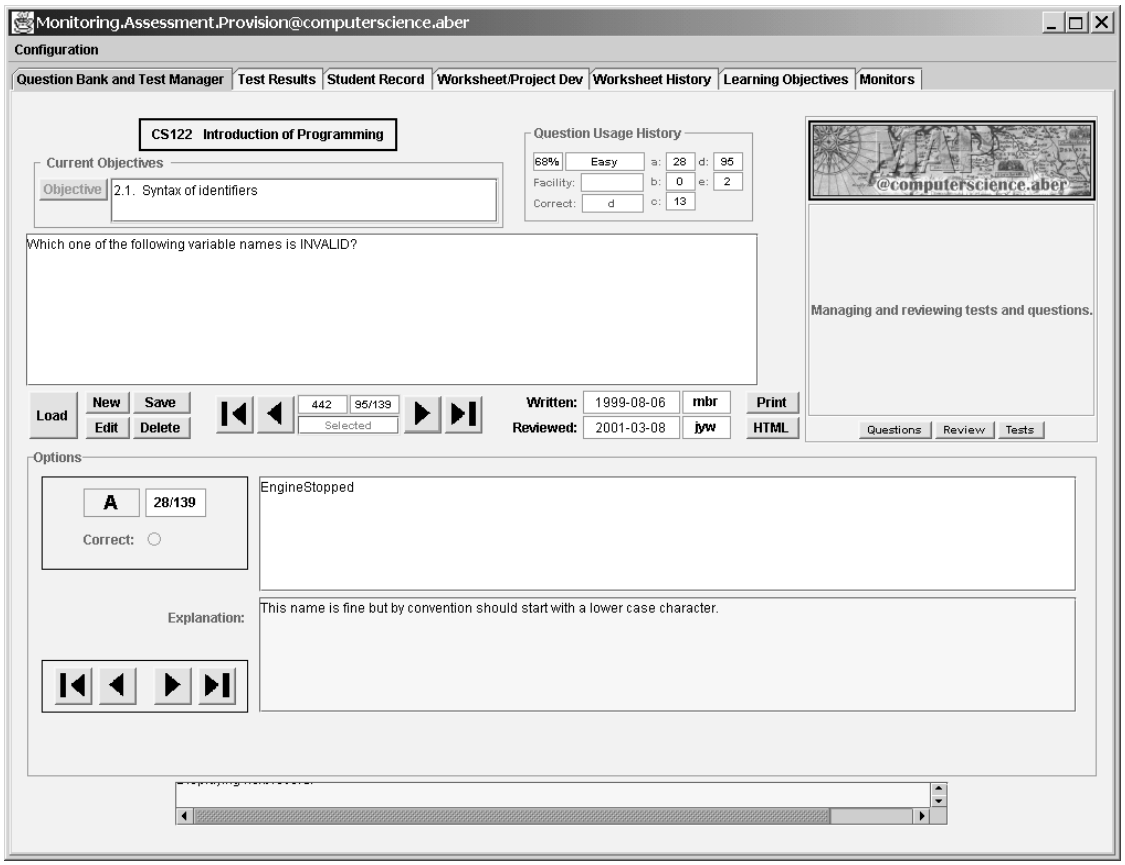


Table 5: MAPView: a listing of the current tabs and brief description of their functions.

Tab	Description
Question bank and test manager:	The user can create, edit and review questions as well as create and publish a test or edit an existing one. Statistics with respect to a question's usage are presented. Questions are explicitly connected to one or more learning objectives for the module.
Learning objectives:	The user can create an hierarchical structure of learning objectives and publish these as a web page.
Test results:	The user can access overall and individual results from a test and generate suitable email shots to the students containing their results and pointers to the learning objectives they need to revisit.
Student record:	The names and photos for students as well as pointers to their test and practical records.
Monitors:	The user can analyze web usage for a given web page. See figures 1 and 2 for screen captures of this interface.
Worksheets/Project development:	The user can create and publish practical worksheets and project assignments.
Worksheet history:	The user can monitor worksheet completion records.

The design process

The changing context of software development: procedural programming and object-oriented programming

There has been a paradigm shift in recent years with respect to programming languages.
Procedural programming defines the task to be modeled as a series of instructions to be carried out sequentially, elaborating with repetition of instructions and alternative paths of execution. Object-oriented programming attempts to solve the task by identifying the entities or objects that populate the world in which the task takes place and defining their behavior and interaction.

It is argued that the latter approach is more likely to yield satisfactory and robust solutions to problems because it more closely models the world as we understand it and therefore can be more readily understood by both client and developer, and consequently more easily manipulated and adapted to changing circumstance.

Historically the most problematic part of commissioning software is the phase of requirements capture and analysis. The client and developer may not share the same descriptions of the world, a client often wrongly assuming that what is given and apparent to themselves will also be given and apparent to the developer. Experience also reveals that because the process of project development has an element of discovery requirements shift: what is required becomes more apparent as the project progresses. Historically the costs for the software development cycle fall predominately in the maintenance phase after delivery of the software: “adaptive” maintenance addresses all those areas you didn’t think of at the time.

Object-oriented development frees all parties concerned from the unrealistic expectation that all can be foreseen. The iterative development of prototypes ensures that there is always a working product that can be used, evaluated and elaborated, keeping the costs of enhancement under control.

In addition, the object-oriented paradigm allows the developer to think at the appropriate level of abstraction. Procedural programming with it emphasis on sequence and procedure more closely mirrors the essential nature of a computer’s hardware and reflects the programmer’s historic concern with conserving resources: computing speed and storage. With the rapid development in both these areas and the consequent fall in costs and improvement in performance the emphasis has shifted way from secondary concerns about performance supported by the underlying infrastructure to the more primary concerns of the task at hand. Modeling at the appropriate conceptual level leads to designs that are simple, readily understandable and robust.

Capturing the design

The Rational Unified Process specifies systems that are user centric (Krutchen, 2000). The first task of the developer is to identify those who will use a system (the actors) and how they will use it (the use cases). The term actor identifies anybody (or, strictly speaking, anything) that gains benefit from the system and describes the rôles people play in the system rather than identifying individuals. It is therefore possible to have the same person fulfilling several rôles, for example as a teacher and as an administrator.

Some of the users identified are: student, question author, question reviewer, test author, test reviewer, worksheet and assignment author, lecturer and demonstrator. More can be added as required.

Some of the use cases identified here might be: “a user authors a question”, another “a user reviews a question” and so on.

Some of the objects and their relationships are identified as: Questions, Tests that are composed of Questions and produce Results, Questions that address one or more Learning Objectives, Results that are processed to produce Emails for feedback to Students.

Finding a design that is extendible

This design will be extendible if we can easily add new objects and flexible if we can revisit existing objects and elaborate them.

During the last academic year a final year student (Regan, 2000) has successfully taken on the task of implementing the use cases involved with the authoring, publishing on the Web and monitoring of practical worksheets. Regan was able to develop independently and integrate “plug in” tabs for these use cases (cf. Worksheet/ProjectDev and Worksheet History, Woodbury, 2001).

Finding a design that is adaptable

Another final year student, Claire Jones, volunteered to revisit the Test object and she has added, among other enhancements, ones that allow us to: record interim results during a test (rather than just at final submission), provide feedback to the test candidate using visual cues to indicate previously answered questions and indicating prior to final submission any questions that have not been answered.

Using MAPView as an exemplar in our curriculum

MAPView is written in Java, the teaching language for our department, and is designed using the project development method (Krutchen, 2000) that we use as an exemplar in our third year module on object-oriented analysis and design.

We use MAPView as a case study in our modules where appropriate to help our students bridge the gap between the experiential and the academic identified by Laurillard as first order and second order learning respectively (Laurillard, 1999) .

Conclusions

The introduction of CAA in the form of objective tests in multiple choice question format has generated much but not universal enthusiasm among students and staff.

We have deployed objective tests in several modules with varying degrees of involvement with other staff, in some instances creating and running the tests, in others creating the tests and letting others administer them, to handing over the software with 10 minutes of instruction and providing on call support.

Objective testing in the frequency we have deployed it has provided timely feedback acknowledged useful by both staff and students.

Students have been involved in the implementation of objective testing throughout, giving us frequent feedback via different mechanisms, and we think that this has reinforced the importance of feedback in teaching in its dictionary sense: that it is a cyclic process (the students gave us feedback on our testing and we gave them feedback on their test performances) and that inherent in the concept of feedback is modification. We were able to adapt to what the students told us about the objective tests and how we administered them in a short enough time frame for them to see results and they in turn had timely and directed feedback in time for them to take remedial action.

Teaching innovation in HE is likely to be a slow process given the historic context in which it takes place(Laurillard 1999), and there will be some staff who greet any attempt to help students as dangerous “spoonfeeding” while many others will exhibit an attitude of “benign indifference”(Ramsden, 2000). Needless to say that in such circumstances funding is sufficient only to provide innovation with a precarious form of life in the interstices between more important things.

We have used the object-oriented paradigm with an appropriate process to develop our software, integrating adaptations or extensions with little effort.

Our experiences with bringing students into the development process for MAPView both directly as developers or indirectly as an exemplar in our teaching have been positive for all parties concerned.

References

Beynon, J. and Mackay, H. (eds) (1993) *Computers into classrooms: more questions than answersI* London: Falmer Press

Beynon, J. and Mackay, H. (eds) (1991) *Technological literacy and the curriculum* London : Falmer Press

Beynon, J. and Mackay, H. (eds) (1991) *Understanding technology in eductation* London : Falmer Press

Bloom, B.S., Engelhart, M.D., Furst, E.J., Hill, W.H., and Krathwohl, D.R. (1956) Taxonomy of Educational Objectives: Cognitive Domain New York: McKay

Carneson, J. Delpierre, G. and Masters, K. Designing and managing Mcq's <http://www.le.ac.uk/cc/ltg/castle/resources/mcqman/mcqcont.html > (25 May 2001)

Friedman, S Morgan, The Inflation Calculator <http://www.westegg.com/inflation> (20 May 2001)

Krutchen, Phillipe (2000) The Rational Unified Process: An Introduction Second Edition London: Addison-Wesley

Laurillard, D (1999) Rethinking University Teaching, a framework for the effective use of educational technology London: Routledge

Negroponte, N (1996) Being Digital London: Hodder and Stoughton

Norman, D A (1988) The Psychology of Everyday Things USA: Harper Collins

Ramsden, P (2000) Learning to Teach in Higher Education London: Routledge

Regan, B (2001) *Enhancements to a Directed Learning Environment* CS39030 (A final year project)

Rust, W B, (1973) Objective Testing in Education and Training London: Sir Isaac Pitman and Sons

Shneiderman, B (1998) Designing the User Interface: Strategies for Effective Human-Computer Interaction Third Edition Harlow: Addison-Wesley

Stephens, D. and Mascia, J. (1967) Results of a Survey into the use of Computer-Assisted Assessment in Institutions of Higher Education in the UK. January 1997 <http:www.lboro.ac.uk/service/fli/flicaa/downloads/survey.pdf> (20 April 2001)

Woodbury, J (2001) Computer Aided Assessment: Computer Science Aberystwyth <http://users.aber.ac.uk/jyw/MAP> (25 May 2001)