Technology-Assisted University Instruction: Large Course Management

William GREENBERG
Dept. of Mathematics, Virginia Tech
Blacksburg, Virginia 24061 USA

and

Michael WILLIAMS
Dept. of Mathematics, Virginia Tech
Blacksburg, Virginia 24061 USA

ABSTRACT

We describe a model of instruction in mathematics which combines the traditional university lecture with computer aided delivery of homework and testing. While this model of emporium instruction was developed at Virginia Tech to deal with the burden of increased class sizes and increasing demands on faculty time, it has, in fact, proven to be an effective pedagogical method with advantages as compared to traditional lecture style instruction. We will argue that it also provides an economic alternative for less developed nations, with particular benefits for the special problems faced by these universities.

Keywords: Emporium, computer education, on-line, test engine, lecture course

1. SETTING

With approximately 20,000 undergraduate students and 5000 graduate students, Virginia Tech is the largest university in the state of Virginia. Because of its large College of Engineering, with more than 5000 students, and because of the requirement that all undergraduate students take mathematics courses, the number of students serviced by the Department of Mathematics is typically in excess of 10,000 students each semester.

The reduction of government support for higher education, which has occurred in Virginia over the past 15 years, has significantly increased faculty teaching-loads. Although the Mathematics faculty numbers about 60 professors and instructors, the burden of teaching so many students has motivated the Mathematics Department to build a *Mathematics Emporium*. This somewhat whimsical term extends the definition of emporium as an open market place: the Mathematics Emporium should be a place where the market of ideas would be freely exchanged among students, faculty and computers. Despite having initially viewed the project as a response to the need to teach very large numbers of students, we have found that teaching mathematics in an emporium style has a number of advantages for the students over traditional lecture courses. [1],[2],[3]

The strategies to be described below were developed for mathematics courses at Virginia Tech. However, it is clear that these Emporium strategies provide a platform which is effective for basic courses in a large number of disciplines outside mathematics. Of even greater significance, the Emporium course can be a cost-effective pedagogical method for university instruction in less developed nations. In fact, such a development (which, by our calculation, can be installed at a cost of $2 per student semester course) would carry additional advantages in such settings: easy transitions to multiple languages, academic comparison and baseline setting with major world universities, easy conversion of applications to relevant local examples, scaling to larger student bodies, etc. We believe the computer emporium can be an essential tool in leveling the playing field between well-funded first world universities and universities in less developed nations.

The Emporium at Virginia Tech contains a large ensemble of computers, located in a now-defunct supermarket building adjacent to campus. A variety of styles for presenting mathematics courses have been developed, in some cases primarily as a result of the aforementioned economic pressures, in some cases entirely because the material itself requires computer analysis, and in some cases because we believe, and shall argue, that such a presentation is actually optimal irrespective of economic and resource considerations.

Some course offerings in mathematics are taught entirely at the Emporium, i.e., there is no classroom component. Other courses have a major segment of the course delivered at the Emporium, and a great number of the remaining courses in mathematics have occasional Emporium assignments. Indeed, at present, 3 mathematics courses with annual enrollment of approximately 4500 students are taught entirely at the Emporium (i.e., *no* classroom component), and half a dozen additional courses handling more than 5500 students each year have major segments of the course taught at the Emporium.

In this article, we will describe an emporium model suitable for the implementation of a combined lecture-based computer-assisted course in mathematics: its structure, the development of necessary software, and its impact on pedagogy.

2. PROGRAM STRUCTURE

Although the Mathematics Department offers several courses which are entirely self-contained at the Emporium [4], we shall describe here a course which retains the traditional faculty lecture

role while depending as well upon Emporium services. We will indicate why such a model is well adapted to meet both changing student strengths and weak-nesses and changing faculty responsibilities.

Approximately half a dozen mathematics courses handling more than 5500 students each year have major segments of the course taught at the Emporium. One of these is vector geometry, required of all mathematics, science and engineering undergraduates. Until recently, this two credit course was taught annually in 40 sections of about 40 students each, meeting twice weekly and requiring, of course, 40 teaching slots. Currently, all 1600 students are taught in 12 sections, each involving one 75 minute weekly lecture and the Emporium component to be described. The need for 12 sections actually reflects the difficulty of finding lecture halls which seat more than 135 students; were larger lecture halls available, the number of sections could be substantially reduced without impacting, in our opinion, the quality of the course.

Unlike on-line courses, those like vector geometry are built around the conventional university lecture format. The material which is presented each week in lecture has, in addition, been placed on-line in a text format, thereby obviating the need to require a course textbook, much to the advantage of the students' budgets. This course, and those like it, have no weekly homework assignment. Instead, the course requires a weekly cycle of practice quizzes, quizzes-for-credit and examinations, each subject to unit deadlines, but otherwise giving students autonomy over their schedules. A student may proceed at any pace he wishes as long as he does not fall behind the default schedule, although most students find it advisable to wait for the weekly lecture, as intended, and then generally have one week to meet the quiz-for-credit deadline.

The weekly practice quizzes and quiz-for-credit are all generated by the same algorithms, so the student is guaranteed that mastering the practice quizzes will enable him to perform at the same level in the quiz-for-credit. Before taking each weekly quiz-for-credit, a student may take as many practice quizzes as desired, but then can take the quiz-for-credit only once. Immediately after completing an attempt at a practice quiz or a weekly quiz-for-credit, the student is shown his score and the correct answers for all problems. Next semester, software will be in place to give the student as well an explanation as to why each problem answered with the wrong answer was solved incorrectly.

The problems used for testing are generated on demand by problem programs. Each problem program is designed to test the understanding of a specific concept covered in the weekly lesson, and must generate hundreds – and in many cases thousands – of variants of the same problem. Since the pool of problems for practice quizzes and the quiz-for-credit is the same, and since the student is encouraged to take the practice quizzes multiple times, the problem programs must be capable of generating a large number of randomly different variations of each problem. Although at the current time the possible answers are presented in a multiple-choice format, the student selects the correct answer from a sufficiently large collection of possible answers – usually between 8 and 12 choices – to effectively eliminate skewing due to guessing.

Practice quizzes and the quiz-for-credit can all be accessed from any Internet site anywhere in the world using virtually any browser by all students enrolled in the course. While some students prefer to take these at the Emporium itself, since on-site resources there include tutoring labs, on-line videos, *under one minute* Emporium support staff response for assistance, etc., the vast majority of these first-year students are now taking both the practice quizzes and the quiz-for-credit from their dormitory rooms or apartments. (We will comment shortly on security aspects).

Periodically students are required to report to the Emporium to take an examination covering the previous three or four weeks of material, and, of course, a final cumulative examination at the end of the semester, subject to failsafe proctoring. Since the testing machinery which produces these examinations is the same as that which produces the quizzes, students are assured that mastery of quiz questions is the *sine non qua* for succeeding in the course.

The key to the effectiveness not only of this sort of course, but, we believe, of most math courses is repetition. The conventional college math course in the United States uses homework in order for the student to master material covered in the lecture. The drawback in the conventional method is that the student typically tries each homework problem once, and at a much later time is advised as to which problems he completed correctly. With the testing machinery in an emporium course, and the encouragement for the student to practice each quiz and test problem as often as he might wish before actually taking the quiz or test for credit, and with instantaneous feedback as to which problems are incorrect, we believe this pedagogical model is not just cost efficient, but actually far more effective than conventional lecture courses.

Initially there were security concerns about allowing students to take quizzes-for-credit from unsupervised Internet locations, because of the temptations to cheat this invariably would create, despite the Virginia Tech Honor Code. As an experiment one semester, the 800 enrolled students in the course were divided into sections which were required to take all quizzes-for-credit at the Emporium (the control group) and sections which were permitted to take quizzes-for-credit from any Internet site (the independent group). To our surprise, the independent group fared better than the control group on the periodic examinations and the final exam, all taken at the well-proctored Emporium. Apparently, the increased flexibility afforded the independent group encouraged them to practice the quizzes more extensively, and, whether or not they were getting assistance while taking the quizzes-for-credit beyond what was allowed by the course rules, the result was that they were learning the material better than students in the control group. Since the totality of the quizzes-for-credit counts only for 17% of the final grade, and the examinations and final count for 83% of the final grade, it is clear that violations of the Honor Code while taking quzzes-for-credit are not likely to affect the final outcome of course grades.

For both on-line courses and these Emporium-assisted courses, a robust test engine is required. A test engine is a system set up to deliver assessments (quizzes, practice tests, etc) to an end user – a student – which also presents the practice tests and examinations to students, grades them, provides the student with access to the results and to correct answers, records, saves, and sends spreadsheet data to the instructor, etc. Each time a practice set, quiz or examination is called up, the student sees a different set of problems. Consequently, the engine has to be capable of delivering large volumes. In addition, these volumes must be delivered in a fashion which is insensitive to the user's platform, and indifferent to the network connection -- dial up, T1 link, Emporium connection, etc. In just the one course under discussion, for example, each of the 800 students per semester takes on average

about 50 practice quizzes, 10 quizzes-for-credit and 4 examinations. The typical quiz has 8 problems, with tests having from 15 to 30 problems. Since each quiz problem must be individually generated for each student each time the problem is accessed, it is evident that the volume is quite massive. In fact, we believe our test engine is the largest such engine in existence.

Because of the very high level of usage, a much richer set of problems is required for this setup than is usually found in a test database. One needs to be sure that students see a different problem each time a problem is accessed. This dictates that the model of problem database that relies on static lists of problems is insufficient, and that problems must be generated from programs which introduce sufficient variety of distinct problems.

To be platform independent at the student end requires use of a common interface that is available ubiquitously. The obvious choice is the web browser. On the server side, we constrained ourselves to open standards and modular design. In this way as course components are improved, and the delivery system upgraded, these changes can be easily implemented. A paramount consideration at all times was to ensure that the system was scalable to large transaction volumes.

3. HARDWARE

These simple design requirements imply that the server consist of a standard web server (we use Apache) and a page delivery service capable of very high volumes of transactions. Apache is public domain, which means it is standards-compliant and free to use. Apache has proven to be extremely robust. In order to retain sufficient programming control to attain these transaction volumes, the most practical environment is java server-pages (JSP). JSP permits a maximum of computational speed and processing in a rich environment. With this architecture, a high-powered machine as server is not necessary. We utilize a pair of aging Sun 3000 servers with 4G of memory and RAID storage. Though they are more than 8 years old, and we have not found the need to upgrade. If we were to replace them at this time, we would use off-the-shelf LINUX systems. One of the machines does the web serving, while the other accommodates the database. Except when we do builds, we rarely see usage exceed 10% of capacity on any machine.

We use Oracle as the database management system. This relational system holds and links data on the students (ID numbers, major discipline, course registrations, email addresses, etc.), on the raw assessments and on the completed exams and quizzes. An instructor's gradebook server regularly taps this database to update this information in gradebook for each student, and passes relevant information to the faculty.

New tests and exams are supplied to the main server through a separate engine that uses Mathematica to generate the individual tests on-demand. As a practical matter we use the test engine to establish caches of new exams in every category. This prevents slowdowns or gaps in service should the test engine server be unavailable, or should the demand for specific quizzes exceed available generation resources. Each of these engines currently resides on one of the two Sun systems or one of the handful of Apple XServes we have available. Because of our modular design, there are no constraints, other than adequate communications links, on where the individual services are located. In this way we gain easy redundancy, guarding against downtime.

4. TEST PROBLEMS

For the types of courses we have developed, the "practice quiz" plays the most fundamental role in the learning process. In the United States, in the traditional lecture format this role is played by the assigned homework problems. For our Emporium courses, this model offers additional motivation to the student to do the practice work, because of its strong correlation with the exams, which are constructed from the same problem programs as the quizzes. This introduces a burden on the problem writer of generating questions of variety and repeatability.

A key feature of the test engine is that it has been made accessible to all of the faculty. We have created an environment where any faculty member can create his own course on-line, with virtually no prior experience in programming. Indeed, we have a number of faculty who are currently involved in placing their courses in the Emporium and whose prior use of computers was restricted to receiving and sending email!

Mathematica, due to its versatility and sophistication, has been chosen as the environment for creating these tests. We have created a series of utilities that simplify the task. The utilities provide a series of commands, functions and shortcuts to compile and display the quiz problems. The instructor needs only to insert the mathematical specifics for each quiz problem into pre-constructed quiz modules, placing, for example, the correct answer in its appropriate location, a series of wrong answers in a corresponding appropriate location, the statement of the problem likewise in its location, etc.

We must be careful not to oversimplify the burden on the instructor. The problems have to be created in a manner that is programmable and allows for distinct versions of the same problem. Moreover, there are many nuances to assuring that duplicate answers are not created, that an apparently wrong answer might be correct for extraneous reasons, that patterns are not unconsciously established among the set of answers which will tip off students as to the correct choice, etc. Moreover, it is desirable to include among wrong answers some which are likely to be derived by common student miscomprehensions. Finally, there is the overriding pedagogical need to assure that the quiz problems teach and test the desired mathematical principles.

Because of the modular construction of the utilities, and their assistance in programming the problems, we have found by experience that mathematics faculty with no prior experience in computer programming can learn to effectively start making quiz problems after one afternoon of preparatory instruction and a selection of model problems.

5. EXAMPLE

We present below an example of a Mathematica program that generates a simple geometry problem, with web browser download including non-randomized answers. The quiz or examination generated on demand for the student has, of course, randomized and indexed answers.

PROGRAM

```
makeproblem["Q10.03",str_]:=Module[{n,k,p,q,r,
x0,t0,tc,nc,nc1,inct1,inct2,incn1,incn2,

ClearAll[x,y,ct,c,v];
```

```
{x0}=ChooseRandom[{1/2,1,2,3,4,5,6},1];
{n,k}=ChooseRandom[{2,3,4,5},2]; p[t_]:=k t;
q[t_]:=(kt)^n; r[t_]:=0; t0=x0/k;
tc=N[(Dot[{p'[t0],q'[t0],r'[t0]},{p"[t0],q"[t0],r"[t0]}]
  )/Sqrt[p'[t0]^2+q'[t0]^2+r'[t0]^2]];
nc1=(Cross[{p'[t0],q'[t0],r'[t0]},{p"[t0],q"[t0],r"[t0]
  }])/Sqrt[p'[t0]^2+q'[t0]^2+r'[t0]^2];
nc=N[nc1[[3]]];
{inct1}=ChooseRandom[{1.65*tc,0.61*tc},1];
{inct2}=ChooseRandom[{2.11*tc,0.48*tc},1];
{incn1}=ChooseRandom[{1.65*nc,0.61*nc},1];
{incn2}=ChooseRandom[{2.11*nc,0.48*nc},1];

ProblemText[str]={{"Problem",{"A point moves along the
curve  "<>tF[y]<>" = "<>tF[x^n]<>"  such that the x-
component of velocity\n is always "<>tF[k]<>".  Find the
tangential and normal components of acceleration at\n the
point with x-component "<>tF[x0]<>"."}},

{"Right",{tF[tc]<>" and "<>tF[nc]<> \r}},
{"Wrong",{tF[tc]<>" and "<>tF[incn1]<> \r}},
{"Wrong",{tF[tc]<>" and "<>tF[incn2]<> \r}},
{"Wrong",{tF[inct1]<>" and "<>tF[incn1]<>" \r}},
{"Wrong",{tF[inct1]<>" and "<>tF[incn2]<> \r}},
{"Wrong",{tF[inct1]<>" and "<>tF[nc]<> \r}},
{"Wrong",{tF[inct2]<>" and "<>tF[incn1]<> \r}},
{"Wrong",{tF[inct2]<>" and "<>tF[incn2]<> \r}},
{"Wrong",{tF[inct2]<>" and "<>tF[nc]<> \r}}  };];

showQuiz["Q10.03"];
```

PROBLEM DISPLAY WITH NON-RANDOMIZED ANSWERS

A point moves along the curve $y = x^5$ such that the x-component of velocity is always 2. Find the tangential and normal components of acceleration at the point with x-component 3.

Right: 2159.99 and 5.33 respectively
Wrong: 3563.99 and 5.33 respectively
Wrong: 2159.99 and 8.80 respectively
Wrong: 4557.59 and 8.80 respectively
Wrong: 2159.99 and 2.56 respectively
Wrong: 4557.59 and 2.56 respectively
Wrong: 3563.99 and 8.80 respectively
Wrong: 4557.59 and 5.33 respectively
Wrong: 3563.99 and 2.56 respectively

A template and a multitude of exemplars from current production programs substantially ease the difficulty of writing new problem programs for the novice. The experienced Mathematica user will notice many functions that are not part of the Mathematica package. These add-ons simplify the recurrent tasks. For example, *makeproblem[ ] := Module[{ }* and *ProblemText[str] = {{"Problem", { }}}* are part of the templates provided to instructional faculty wishing to author quizzes.

A universal course file holds all the quiz problems for the semester course. The preamble of the file contains instructions for choosing problems in order to create practice quizzes, quizzes-for-credit, examinations, and the final exam. This Mathematica file, typically about 500 KB, contains all of the testing material for a single course. A simple java compiler has been written to generate actual quizzes in java code, which are then archived on the Emporium workstation. Once the quizzes have been compiled, there is no further need for Mathematica: the archive contains only java versions of the tests. Because we have courses with more than a thousand students, all of whom have access to the Emporium workstation both at the Emporium and, in most cases, from their lodgings via the Internet, we do not expect the workstation to keep up with demand by generating java code on demand. Therefore, we generate these tests in bulk before they are needed in the corresponding course. For example, in a typical course with 1000 students, we might generate 6 to 8 thousand practice quizzes (for each of 10 quizzes) and 1100 exams (for each of 4 exams). The entire java output for the entire course can be contained on 3 or 4 CD's.

It is important to realize that the Emporium does not require an Internet connection (although such a connection will of course yield many other benefits). The entire testing system is self-contained, if students use the Emporium for their computer access. This is likely to be of importance in some less developed nations.

6. ECONOMICS AND CONCLUSIONS

The conversion of traditional classroom course offerings to models of the sort described in the preceding has substantially reduced costs, as well as improving staff working climate and morale, primarily by reducing the demands on senior lecturers in so many classrooms. Furthermore, when the performance in later courses of students who have completed courses under the new models are compared to those who learned the material under the traditional model, we find that the former significantly out-perform the latter. We believe this is due to increased time-on-task and an increased independence [5].

More important, the Math Emporium has permitted us to make a discontinuous change to the methods of operation of our course offerings. We have used this unique asset to transform traditional classroom-based courses into more efficient technology-based learning programs. The structure of the program is very different from the conventional one, having a new set of expectations and motivations. To summarize, the results include

:• Students are more self-reliant
• Students budget their time more effectively and satisfyingly
• We enjoy a substantial cost savings
• Students demonstrate increased proficiency in downstream course work
• We have learned much about distance learning
• A new fully automated testing system broadens the impact of testing
• A stream-lined process takes source material into structured web resources effectively
• The model we are using scales well to large enrollment courses
:• The model is portable and easily translates to other languages
:• Professors with minimal programming experience can modify or develop quiz content, as desired.

Finally, we emphasize that the system outlined retains the traditional lecture format for the university course: it is not an online long distance course.

7. REFERENCES

[1] Quinn, F and Williams, M., "Lessons from the Emporium 1: Goals and economics," Preprint November 2003, at http://www.math.vt.edu/people/quinn/education

[2] _____, "Lessons from the Emporium 2: Help for computer-based learning," Preprint November 2003, at http://www.math.vt.edu/people/quinn/education

[3] _____, “Lessons from the Emporium 3: Testing and course design,” Preprint July 2004, at http://www.math.vt.edu/people/quinn/education

[4] Greenberg, W. and Williams, M., “Reform in the Teaching of Mathematics at the University Level: the Computer Emporium,” in *Proceedings of the RAMAD International Conference on Environmental Modeling and Simulation*, Dakar, 2004.

[5] Williams, M., "The Math Emporium: The Changing Academy or Changing the Academy," in *Developing Faculty to Use Technology*, David Brown, ed., Anker Publ., Boston 2003, 285-287.