TEACHING AND LEARNING PORTFOLIO

by

Lisa Torrey

Last Updated: May 2006

This portfolio submitted in partial fulfillment of the requirements for the Delta Certificate in Research, Teaching, and Learning.

Delta Program in Research, Teaching, and Learning
University of Wisconsin-Madison

The Delta Program in Research, Teaching, and Learning is a project of the Center of the Integration of Research, Teaching, and Learning (CIRTL—Grant No. 0227592). CIRTL is a National Science Foundation sponsored initiative committed to developing and supporting a learning community of STEM faculty, post-docs, graduate students, and staff who are dedicated to implementing and advancing effective teaching practices for diverse student audiences. Any opinions, findings and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation.

For more information, please call us at 608-261-1180 or visit http://www.delta.wisc.edu.

2006

Teaching and Learning Portfolio

Lisa Torrey

May, 2006

Contents

Overview

This portfolio is a forum for the expression of my ideas about teaching and learning, and also a collection of evidence from my experiences in putting those ideas into practice. It is a working document that evolves as my thoughts do. Communicating these thoughts and reflecting on my experiences helps me to develop as a teacher. I hope it will also demonstrate to you the kind of teacher I have been and will be.

Curriculum Vitae

Lisa Torrey

Education

University of Wisconsin-Madison: Madison, WI (2003–present)
Master's degree in Computer Science in May 2005
Working towards a Ph.D.

Dartmouth College: Hanover, NH (1999–2003)
A.B. in Computer Science modified with Engineering, Summa cum Laude

Teaching Experience

Instructor: University of Wisconsin-Madison (2003–2004)
Taught introductory computer science courses in Java. Planned and gave lectures, helped develop assignments, assisted students and graded their work.

Section Leader: Dartmouth College (2001–2002)
Conducted weekly discussion sections for students in an introductory computer science course. Assisted students and graded their work.

Writing Assistant: Dartmouth College (2000–2001)
Edited student papers and worked with first-year students to improve their writing.

Research Experience

Research Assistant: University of Wisconsin-Madison (2004–present)
Contributed to research on reinforcement learning applied to complex multi-agent worlds. Investigated topics like concept transfer and advice-giving.

Student Intern: Sandia National Laboratories (Summer 2005)
Conducted research applying machine learning to network intrusion detection. Designed programs to detect coordinated network attacks.

Student Intern: Structural Informatics Group, Seattle WA (Summer 2002)
Designed user interfaces to explore medical database information through text, images, and 3D scenes, as part of the CRA Distributed Mentor Project.

Service

Delta Steering Committee: University of Wisconsin-Madison (2005–2006)
An NSF-funded program for improving undergraduate science education.

Affiliations

Member of AAAI since 2005
Member of SIGCSE since 2006

Honors

Phi Beta Kappa, 2003
High honors for undergraduate thesis in computer science, 2003
Honorable Mention, CRA Outstanding Undergraduate Award, 2003
Golden Key Society, for college students in the top 15% of their class, 2001
National Society of Collegiate Scholars, for service and achievement, 2000

The Delta Program in Research, Teaching, and Learning

While I was a graduate student I participated in the *Delta Program in Research, Teaching and Learning*, a learning community for graduate students, post-docs, academic staff, and faculty at the University of Wisconsin. Its purpose is to improve student learning in science fields through professional development for current and future faculty. The Delta website is at http://www.delta.wisc.edu .

The three central themes of the Delta program are:

- Teaching-as-research: *"The deliberate, systematic, and reflective use of research methods to develop and implement teaching practices that advance the learning experiences and learning outcomes of students and teachers."*

- Learning community: "*Bringing together groups of people for shared learning, discovery, and the generation of knowledge.*"

- Learning through diversity: "*Drawing upon the diversity of students to enhance and enrich learning for all.*"

The Delta Teaching and Learning Certificate

Through this program I earned the *Teaching and Learning Certificate*, which recognizes the completion of a program of study with Delta. Its requirements are:

a) Two graduate courses on teaching and learning.

I took the following two courses:

1) *Effective Teaching With Technology*. We experimented with existing technological teaching tools and developed our own, conducting teaching-as-research projects into their effectiveness.
2) *The College Classroom*. We developed teaching philosophies, created instructional materials, and taught short lessons to a community of peers.

b) Participation in the Delta Learning Community.

I participated in the program *Creating a Collaborative Learning Environment*, where I met with a small group of faculty and graduate students to discuss readings on teaching and learning.

c) A teaching and learning internship.

I worked with an environmental engineering professor to improve her students' learning of chemical reactor models (see Appendix for details).

d) Development and presentation of a teaching and learning portfolio.

Teaching Philosophy

"In my experience we tend to be most interested in challenges just at the edge of our comfort zone, which is ideal for learning."
– page 5

"No single person can create a community, but teachers are often in a position to create an environment in which it can develop."
– page 5

"To solve new problems, though, students need insight and creativity. I am not certain these qualities can be taught, but I think perhaps they can be honed with exercise, like muscles."
– page 6

Teaching Philosophy

I belong to a community of scholars who seek to make machines learn. Perhaps machine learning seems premature when so much about human learning remains a mystery to us. On the other hand, the human mind is more complex than any machine. I believe the two learning disciplines can inspire each other and grow together. In this way, I see my teaching and my research as intersecting parts of a single learning discipline.

I was drawn to both teaching and research in computer science because there are so many important problems that the field has such potential to help solve. How can we maintain security over large-scale wireless networks? What secrets can we find within our growing stores of genome data? Can we make a computer communicate effectively in a natural language? Through teaching, I hope to attract, retain, and train people who will tackle these kinds of problems.

Attracting students: Problem-based learning

I see introductory courses as the key to attracting students to computer science. In theory these courses exist to provide students with knowledge and skills for higher-level courses, but in practice they also serve as advertisements for the entire field. If we wish to be truthful in advertising, I believe we should design introductory courses to reveal the excitement and potential of the field.

I try to do this by teaching students using real, relevant problems. These can be inspired by past or present research questions, but they need not be original – just new to the students. They can also be inspired by students' interests. For example, I have asked students in a programming course to suggest ideas for programs they could use in their own lives. From their ideas, I designed exercises that connected programming with solving real-life problems. (See artifact #2 for an example.)

For students to learn the most from problem-solving, I believe the problems need to be challenging at just the right level. Ideally, I would design exercises with a range of difficulty and encourage students to choose what interests them most; in my experience we tend to be most interested in challenges just at the edge of our comfort zone, which is ideal for learning. In practice, demands on our time influence both my designs and students' choices, but I continue to work towards that ideal.

Retaining students: Learning communities

I believe that community plays a large role in retaining students through higher-level courses and advanced degrees. Students who feel like they belong to an energizing and supportive academic society will persevere and learn better than students who see education as a solitary commitment. No single person can create a community, but teachers are often in a position to create an environment in which it can develop.

I try to do this by bringing students together to face shared challenges. For example, I have put students in small long-term groups for the semester. They sat together, worked together for in-class exercises, contacted each other if they missed a class, and met to study or do homework if they chose. Before I tried this, I found that many students never even learned each others' names. With these base groups, even the shyest students had a chance to form connections. (See artifact #3 for some reflection on this experience.)

There are as many potential pitfalls in group work as there are benefits. Ideally, students will correct each others' misunderstandings, contribute according to their abilities,

and learn with a broader perspective than they would alone. In practice, I find that carefully designed assignments and evaluation policies are needed to help groups function positively. These are skills I hope to improve as I gain more teaching experience.

Training scientists: Discovery learning

Preparing students for careers in computer science certainly includes giving them a solid foundation of knowledge and skills. To solve new problems, though, students need insight and creativity. I am not certain these qualities can be taught, but I think perhaps they can be honed with exercise, like muscles. I believe that students who learn actively, rediscovering material rather than passively absorbing it in neat packages, develop into better creative problem-solvers.

I hope to do this by presenting problems without immediately introducing solutions; by sparking ideas from students and calling on their knowledge and intuition to develop those ideas, instead of giving them a complete solution to memorize. For example, in an artificial intelligence course, I could ask students how we might adapt what we know about neurons in the brain to design an artificial neural network. With prompting, their intuition might carry the idea quite close to the backpropagation algorithm, and I believe they would retain a better understanding of the concepts than if I threw equations at them first. (See artifact #4 for an example.)

If we truly want students to master something, common wisdom says we need to put it on the exam. This has been true in my experience; students are adept at calibrating their efforts to the requirements we set. If I want students to become creative problem-solvers, I will need to create assessments that require them to solve problems creatively. Doing so without evoking students' fear of failure, which hampers creativity, is challenging. As I gain experience, I hope to find this balance.

Beneath it all: Teaching-as-research

One underlying methodology supports every teaching technique that I try. This methodology has roots in research, and is known as *teaching-as-research* or *scientific teaching*. I was introduced to it by a professional development program in teaching and learning, and it quickly became an important practice for me.

Teaching-as-research is the idea that we can evaluate teaching objectively, judging it by its measurable effects on student learning, in the same scientific way that we measure other complex processes. Rather than relying on subjective and anecdotal evidence, I can systematically measure what my students learn when I use a technique and then modify the technique accordingly. (See artifact #1 for an example.)

This is not always easy to do, since a classroom is not a laboratory. I would not want to make one group of students a control group, teaching them in a way that I think is less effective. This means I need to try to measure student learning constantly and keep records so I can use the past as a control group. In the short term, this can give me an ongoing summary of what my students understand, which I can use to address gaps in their understanding. In the long term, it can guide me in improving purposefully as a teacher.

The next section of this portfolio contains reflections on issues like these, and displays several artifacts that demonstrate the concepts in practice.

Reflections and Artifacts

"We were able to measure differences in performance and conclude that the tutorial made statistically significant improvements in some of their problem-solving skills."
– page 8

"We do not tend to think of group work as something to be taught, but it is a set of skills that we could at least discuss with students."
– page 10

"I start with what the students know, and guide them in using their intuition and reasoning to approach the problem."
– page 15

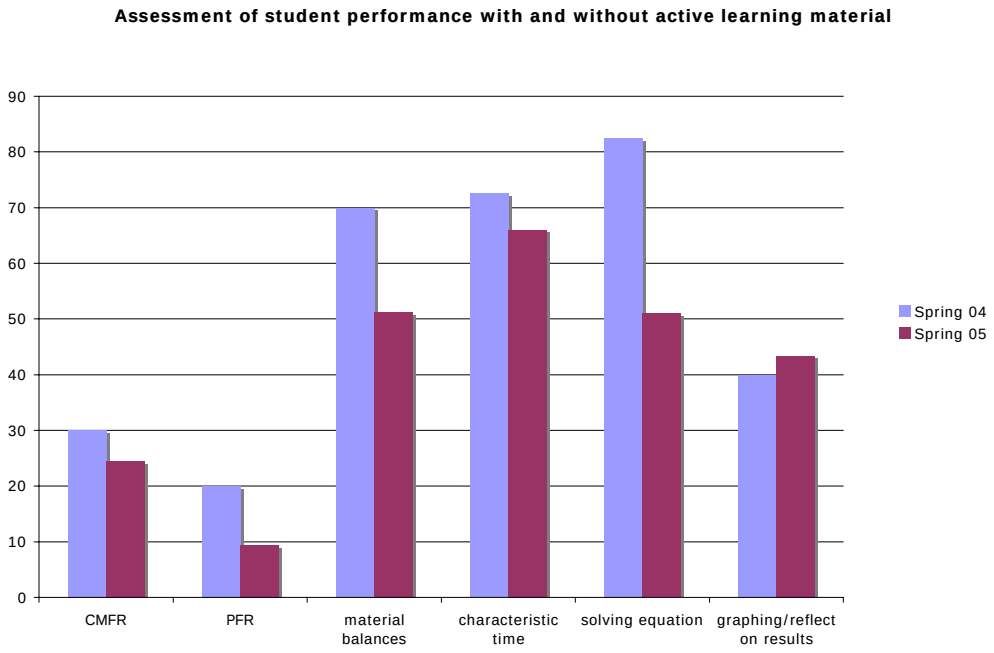
"If I design accommodations so that they benefit everyone, then diversity becomes an asset, not a burden to bear."
– page 18

Teaching-as-research

The first time I taught a course, I naturally borrowed the techniques my own teachers had used. The second time I taught, I incorporated some new ideas that I hoped would improve my students' learning. Did they do so? I have only anecdotal evidence, so the best I can say is that I hope they did. As I built this portfolio based on these early teaching experiences, I often wished I had more direct evidence of student learning.

The first tenet of the Delta program is that teachers can do more than rely on anecdotal or subjective evidence to evaluate and develop teaching techniques. We could instead treat the learning process with scientific rigor, gathering objective evidence about which techniques work better. This process, as I described in my teaching philosophy, is *teaching-as-research*.

I had the opportunity to practice teaching-as-research in my Delta teaching and learning internship. I worked with an environmental engineering professor on a project to improve her students' learning of chemical reactor models. We designed an online tutorial on the subject, had the students complete it, and gave them a quiz that had also been given in a previous semester. We were able to measure differences in performance and conclude that the tutorial made statistically significant improvements in some of their problem-solving skills, as shown in the figure below.



Artifact #1 – Percent of students struggling with concepts.

Detailed information on this project is included in the Appendix, along with a related project that I undertook for a course on teaching with technology.

Problem-based learning

In my teaching philosophy, I presented my opinion that introductory courses should contain problems that are real and relevant to students. I do not mean the textbook version of a real problem, which tends to be a contrived problem dressed up in an imaginary scenario. For a problem to be truly relevant, I think its solution needs to be of actual use to students. In a programming course, for instance, a truly relevant assignment would produce a program that students would later use.

In one course, I asked my students to suggest such programs. One student said he could use a program to quiz him on his Spanish vocabulary, and the Java programming exercise below is the result. I gave this exercise as an in-class activity. I hope that it improved that student's Spanish as well as his programming skills.

In future programming courses, and perhaps in other types of courses if it seems appropriate, I plan to continue using this strategy. I may require students to give me one or two ideas each, since only a few gave them voluntarily. Not every idea may be usable; my first priority is to design reasonable exercises that cover the concepts they need to practice, and some ideas may not fit in.

Artifact #2 – Programming Exercise

```
In this exercise, you will write a program to quiz you on vocabulary in
a foreign language. The vocabulary will come from an input file that
looks like the following:

spring:ethoie
summer:laer
autumn:iavas
winter:rhiw

... and so on. Each line contains an English word and its translation
into the other language. (The other language in this case happens to be
Sindarin, which was created by J.R.R. Tolkien for his books.)

Your program should read the words and put them into two arrays. Then
it should loop through the English array and do the following:

      - print the English word and ask for a translation
      - read the word the user types
      - compare this to the translated word, ignoring case
      - tell the user whether it's right or wrong
      - give the user 3 chances to get it right before giving the
        answer and going on to the next word
```

Learning community

In my teaching philosophy, I discussed the role that I feel a supportive community can play among students. In one course, I put these ideas into practice by placing students in small long-term groups for the semester. They sat together, worked together for in-class exercises, contacted each other if they missed a class, and met to study or do homework if they chose.

On the following pages are two evaluation forms from students in this course. They comment on the strengths and weaknesses of the group work they did. I chose these particular forms because they represent two ends of the student spectrum: student A appears to be very comfortable with the course, while student B appears to be struggling.

Student A confirmed a few things that I had hoped for: that the groups helped students get to know each other, and that group members reached out to each other for help outside class. Student B describes how working with a group makes it less likely that individual students will get stuck in the problem-solving process. By working together, the group made more progress than Student B could have made alone.

Both students bring up concerns that I will need to address in the future. Student A admits that groups sometimes took shortcuts instead of doing exercises the way I had intended. I plan to prevent that in the future by collecting any group work they do in class, even if I do not intend to grade it. Student B points out that if some group members work much faster than others, the slower members can get left behind. In the future I plan to make sure group members do not have large disparities in experience or speed by giving an exercise to determine their abilities before assigning groups.

In courses where group work is a substantial part of the course (which it was not here), I might go further and think about training students to work well in groups. We do not tend to think of group work as something to be taught, but it is a set of skills that we could at least discuss with students. I plan to communicate my expectations, and perhaps to have students evaluate their group members according to those standards.

I also believe that a learning community can play a large role for faculty. While research disciplines thrive on strong communities and the exchange of ideas, teaching is often performed nearly in isolation. Since joining the Delta program, I have been reading about and discussing teaching almost on a weekly basis. I find teaching ideas occurring to me so frequently that I started a text file on my computer to keep track of them all. Now I can hardly imagine teaching in isolation; I will always look for an active learning community to stimulate my teaching.

Artifact #3 – Student A evaluation, page 1

Evaluation Form

1) Do you do the reading as it's assigned? If so, how well do you understand what you read?

I usually keep up on the reading, but not always. I generally understand the reading quite well, but I still appreciate going over it again in lecture. The multiple perspectives are very helpful.

2) When you presented a section to your group, how did that affect your understanding and/or memory of the material in that section?

My group quickly fell into a pattern of taking notes and copying them for other group members, so I didn't get much out of presenting to the group, but I think I would have had we done it the way you intended.

3) How about when another member of your group presented?

This depends heavily on whether I had done the assigned reading. When I had, the presentations didn't do much for me. When I hadn't, however, I found that later, when I was doing the reading, I had to ~~[illegible]~~ take much ~~less~~ fewer notes.

4) How useful is it to get to know a group and work with them during class? Does it make the long class time any less boring?

I wouldn't say it's affected how boring the class is (nor would I say the class is boring), but it has definitely allowed me to get to know those students better, which is always a good thing. I also find that group members are more likely to reach out to others in the group for out-of-class help.

5) How well do you understand lectures? What makes you get lost, and what could make them better?

I understand lecture very well and rarely get lost. I think having more classes in lab with interactive lecturing would be a great improvement—sometimes it is hard to understand the implications of the concepts out of context, and watching you code is not nearly as helpful as having to apply it ourselves ~~[illegible]~~ would be.

Artifact #3 – Student A evaluation, page 2

6) How useful are the in-class exercises and lab exercises? Why?
As I suggested in 5), the lab exercises are very useful, because it is the only chance we have to experience guided application. The in-class applications are also useful, but they don't hold a candle to the lab work.

7) How useful has CodeLab been to you? Why?
I don't like CodeLab exercises because their complete lack of context makes them seem trivial to me. However, they are similar to the exam questions, so they are definitely useful.

8) How useful have the assignments been to you? Why?
The assignments have been more useful for the exact reason Codelab has not: context and application. However, I am still waiting for and looking forward to assignments where I have to think and design rather than just follow very detailed directions.

9) What things were most helpful in preparing you for the exam? Why?
Codelab and reviewing the notes, since these were the most related to the exam format.

10) How could this class be made more interesting?
Interactive lecturing in the lab (see 5).
It also will be when we know enough to do more complicated stuff.

11) Are you a somewhat experienced person, or just starting to learn programming? (Just to calibrate the responses.)
Somewhat experienced

Artifact #3 – Student B evaluation, page 1

Evaluation Form

1) Do you do the reading as it's assigned? If so, how well do you understand what you read? I always do the reading, either before or after. But sometimes dont understand.

2) When you presented a section to your group, how did that affect your understanding and/or memory of the material in that section?
I dont think it help me all that much.

3) How about when another member of your group presented?
It help a little because then I could ask them questions.

4) How useful is it to get to know a group and work with them during class? Does it make the long class time any less boring? It helps because if I dont know how to do something the someone else will and I won't be staring blankly formost of the period

5) How well do you understand lectures? What makes you get lost, and what could make them better? Termonolgy and program, Sometime don't explain what you are doing and you speed through it

Artifact #3 – Student B evaluation, page 2

6) How useful are the in-class exercises and lab exercises? Why?
No that much, because it take me longer to think and understand program then any one in my group.

7) How useful has CodeLab been to you? Why?
Very, because it gave feedback right a way

8) How useful have the assignments been to you? Why?
A tittle because I did not really understand then and just did a half job

9) What things were most helpful in preparing you for the exam? Why?

10) How could this class be made more interesting?

11) Are you a somewhat experienced person, or just starting to learn programming? (Just to calibrate the responses.)
just learning.

Discovery learning

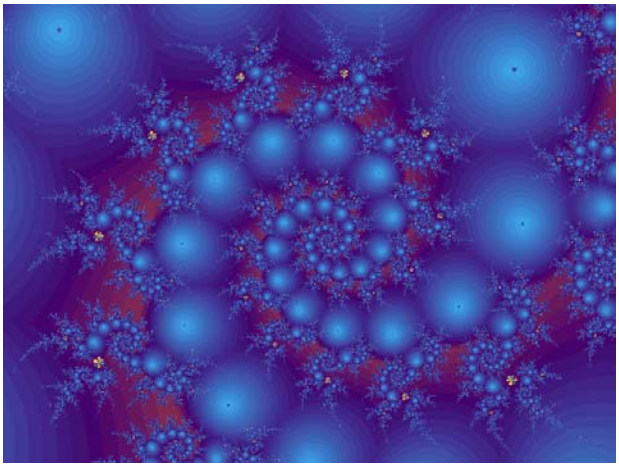
In my teaching philosophy, I discussed ideas for training students' creative problem-solving skills. I believe a good approach is to treat problems I use as examples in class as new, not as neatly packaged material from some all-knowing textbook. I start with what the students know, and guide them in using their intuition and reasoning to approach the problem.

As an example, on the following pages is a lesson plan that I created and used to teach some of my favorite topics in introductory computer science: searching and sorting algorithms. These are often the first non-trivial algorithms that students see, and they can be difficult for new students to master.

I begin by asking them to describe the strategies they use themselves to sort playing cards as algorithms. Their descriptions always correspond to standard algorithms like binary search, selection sort, and insertion sort. When they realize this, they become confident that the topic is actually familiar to them, and their intuition is relevant.

I continue to use the playing cards to demonstrate mergesort and quicksort. One student starts with all the cards, splits them in two piles and passes the piles to two other students; when each student has one card, they reverse the procedure to merge them.

Besides the physical prop of playing cards, I also use visual imagery. I show some fractal images, like the figure below, to illustrate recursion. To show students how much running times really matter, I use a Java applet that has sorting algorithms compete; they enjoy seeing how thoroughly quicksort leaves the others in the dust. These displays are important for students with visual learning styles and enjoyable for all.



A fractal image.

In higher-level courses, physical props like cards will not be necessary for demonstrating algorithms. Still, I believe starting with what students know will give them a stronger grasp of new concepts and teach them how to approach new problems.

Artifact #4 – Lesson plan, page 1

```
*****************************************************************************

Searching: find a number in an array

       1) How do you find the ace in these 10 cards?
              * called linear search: best case 1, worst case 10, avg case 5
                     - this is O(n)

       2) What if I told you they were sorted?
              * called binary search: best case 1, worst case 4, avg case 2
                     - this is O(log n)

Sorting: get the numbers in an array to be in order

       1) Take these cards. how did you sort them? give me the steps.
              * write down algorithms

              A. Selection sort:
                     while (not sorted)
                       select the smallest unsorted card
                       put it in its place

              B. Insertion sort:
                     for (each card)
                       find its place in the sorted list so far
                       insert it there

       2) How do we figure out the efficiency of these?
              * an operation is one comparison
              * selection: n+(n-1)+(n-2)+...+0
              * insertion: 0+1+2+...+n
              * they have the same worst-case formula: n(n+1)/2 -> O(n^2)
                     - but selection always takes this long, no matter what
                     - insertion is sometimes faster, depending on starting order

       3) Other ways to sort (that a human wouldn't use)

              A. How would you sort two groups that were already sorted?
                     * this is a MERGE
                     * demo of mergesort: 8 cards, need 6 people
                            - split in half and pass halves on
                            - when you have just 2, sort them and pass back
                            - when you get some back, MERGE them together
                     * this is recursion!
                            - a method calls itself to solve a smaller problem
                            - each person does same instructions with smaller set
                            - each person is another call of the method
                            - eventually there's the "base case" (smallest problem)
                            - then the calls return back up until they're all done
                     * how efficient is it?
                            - draw the tree: n * log n -> O(nlog n)  (better)

              B. You can split around a "pivot", smaller vs. larger
                     * demo of quicksort:
                            - choose pivot with your eyes closed and split
                            - keep pivot and pass halves on
                            - when you have just 2, sort them and pass back
                            - when you get some back, put on either side of pivot
                     * recursion again; same sort of tree, so also O(nlog n)
                            - but what if the pivot was always really bad? O(n)
                            - almost always it isn't, though
                            - and when it isn't it's faster (less copying)

       4) Watch the computer demo!
```

Artifact #4 – Lesson plan, page 2

```
*****************************************************************************

Recursion: method calling itself on smaller problems until base case
       - has to stop somewhere, or you get infinite recursions
       - since each call gets its own stack frame: stack overflow

       1) Fractals!
              - these are the coolest way to look at recursion
              - zoom in on part and it looks the same: smaller version
              - fractals are mathematically infinite, but pixels aren't,
                so fractal images have base cases too
              - show pictures

       2) Mystery method: what does it calculate? (n!)

              public int mystery (int n) {
                     if (n==0)
                            return 1;
                     else
                            return (n * mystery(n-1));
              }

              * draw the tree for n=5 to see how it works

       3) Recursion is not always the best way-- sometimes really inefficient
              * fibonacci sequence: 1 1 2 3 5 8 13 21
              * what's the pattern? fib(n) = fib(n-1) + fib(n-2)
              * what's the base case? there are two: fib(0)=fib(1)=1
              * here's the method:

                     public int fib (int n) {
                            if (n==0 || n==1)
                                   return 1;
                            else
                                   return fib(n-1) + fib(n-2);
                     }

              * draw the tree for n=5; what's silly about this?
              * recursion is a bad way to do this, b/c it repeats work a lot
              * would be better to do this with a loop
              * use recursion when it makes things more efficient or when
                iteration would be really hard to program

*****************************************************************************
```

Student diversity

I first heard the story of "curb cuts" in one of my Delta courses. Curb cuts are the small ramps that create smooth transitions from the sidewalk to the street. They were made mandatory by the Americans with Disabilities Act of 1990 so that people in wheelchairs could navigate sidewalks. By now, since people also use them for strollers, bikes, roller blades, and so on, it is clear that curb cuts are just a good idea in general.

Like people on a sidewalk, learners come with a wide diversity of abilities, backgrounds, and preferences. Many will succeed in almost any learning environment, but for some, an accommodating environment can make all the difference. But remember the curb cuts: if I design accommodations so that they benefit everyone, then diversity becomes an asset, not a burden to bear.

There are several accommodating practices in which I believe strongly. One is untimed exams – these allow students to work without undue pressure, and to show me what they can actually do, rather than what they can do in an artificially short period of time. Another is "late days" for homework – when each student automatically receives a small number of one-day extensions on assignments, to be used or not as the student wishes.

These policies are based on the standard accommodations usually given to students with learning disabilities or special circumstances. However, I see them as academic curb cuts that can benefit all students. By expanding them to apply to all students, I use ideas born from diversity to improve learning for all.

On the following pages is a syllabus for a hypothetical upper-level course in machine learning. It provides examples of these ideas, and also reflects other ideas I have discussed in this portfolio. I hope that it also provides a glimpse of the kind of teacher I will be in the future.

Artifact #5 – Syllabus, page 1

Syllabus: Machine Learning

Instructor: Lisa Torrey

Office hours: *fill in*, or by appointment

Textbook: *Machine Learning* by Tom Mitchell, McGraw-Hill 2000

Course overview

Machine learning (ML) is a problem-solving approach for prediction problems that are too big or poorly understood to solve directly. Machine learning algorithms can help us understand complex mechanisms and create models to predict their behavior.

For example, no one really understands the visual mechanisms that allow us to recognize human faces in photographs. How can we train a computer to do so if we don't know how we do it? Machine learning is a way to approach problems like these.

Course goals

- To interest you in the possibilities of ML for problem-solving.
- To prepare you to apply it for problem-solving where appropriate, either by using and adapting known algorithms or by designing your own.
- To make you confident about doing this without expert guidance. For students focusing on ML, this means laying the groundwork for becoming an expert. For students whose focus lies in other areas, it means preparing you to use ML techniques in your work without having to say "I'll leave this part to the ML experts to figure out."

Course structure

- During class we will discuss the topics at a conceptual level, introducing you to the major ideas in machine learning.
- In homework assignments, you will implement some algorithms.
- When necessary, there will be reading assignments to prepare you for a class discussion.
- There will be a final exam that pulls all of these elements together into a coherent whole.

Evaluation

- Homework assignments – 60%
- Reading assignments – 10%
- Final exam – 30%

Homework Assignments

For most of the homework assignments, you will be able to choose an algorithm to implement. I do this to allow you to focus on things that interest you.

You will have five (5) *late days*, each of which you may use to delay a homework deadline by 24 hours. You can "spend" these days all on one assignment (not recommended), or spread them out, or not use them at all, as you choose. I do this to give the course some flexibility for everyone if it's needed. If you use up all your late days, I will no longer accept late assignments from you.

I will give you feedback on your homework assignments, and you will be able to revise them accordingly. I do this because the revision process is a good learning experience. Each homework can be resubmitted at most once, and is due the class period after you received the feedback.

Artifact #5 – Syllabus, page 2

Reading Assignments

I may assign short reading assignments before some class discussions, along with a few associated questions that you will answer using the online system at [*web link*]. These questions will lead us into the discussion and help me adjust to your background knowledge level.

Final Exam

You may bring to the exam 1 reference page with anything you want *handwritten* on it. I do this so you do not need to memorize things like formulas, and because creating the page is also a good learning experience.

The exam will be untimed, so that you can work without being rushed.

Expectations

Academic Honesty Policy: I trust you to follow the University's academic honesty policy, details of which are at [*web link*], in all your work in this course. It is important to me that you learn well in this course, which is only possible if you do your work diligently and honestly.

You may help each other with homework, but you must personally write everything you hand in. If you ever wonder whether you are getting too much help, ask yourself this: will you be able to explain to me how every part of your program works? If not, then you are overstepping the bounds of academic integrity.

Attendance and participation: Your participation is what makes class time worthwhile. I do not take attendance, and I do not expect you to give me an excuse for absences. However, I do expect you to contact me or a classmate to get any information you miss, and I will contact you if I become concerned that chronic absence is affecting your learning.

Office hours: I would very much like to talk with every student face to face at least once during the term. Please make a point to come to my office hours at least once.

Concerns, Conflicts, and Disabilities: If you have any concerns, disabilities that might affect you in this course, or known conflicts with the requirements, please come talk to me during the *first two weeks* of the course so that we can work out any necessary accommodations.

***Feedback*:** I request and welcome feedback from you about the course. If you have comments or suggestions about any part of the course, please share them with me. I will provide several opportunities for anonymous feedback, but you should not feel limited to those.

Topics to be covered

- Supervised classification – nearest neighbor, decision trees, perceptrons, rule-learning, neural networks, support vector machines
- Issues in supervised classification – inductive bias, noise, overfitting, Occam's razor, feature selection, dimensionality, local minima
- Assessing algorithm performance – accuracy, precision/recall, *t*-tests, the PAC and mistake bound models, VC dimension
- Alternate learning settings – unsupervised, semi-supervised, active, explanation-based, reinforcement, generative models, first-order
- How ML fits into the rest of AI, and where it might go in the future

Appendix

"Grading is important in the current higher education system, but it should always take second place to learning."
– page 23

"They could follow and understand a problem solved in class, but their work on tests demonstrated that they had difficulties doing so on their own."
– page 24

"They needed a learning opportunity that was more difficult than lecture, but less difficult than the large assignments, and they needed it before the first exam."
– page 29

Grading Philosophy

Grading is important within the current system of higher education primarily because employers, graduate school admissions officers, and others demand quick ways to compare students. By providing an easy means of ranking students, grades fulfill an external demand. Also, as they progress through schooling many students become dependent upon grades; they begin to complete their work partly for the reward of high grades or for fear of low grades. Grading is therefore a mechanism for both student evaluation and student motivation.

I find this mechanism to be far from ideal. Even in the best circumstances it is a primitive tool, and in the worst circumstances it can be harmful to student learning. I will explain why I believe this to be true, and I will describe my approach to grading, which attempts to meet the practical demand for grades while minimizing their potential negative impact.

Grading is a primitive tool for evaluation because it provides almost no information about students' actual abilities. Since a grade is typically an average of performance, students with entirely different abilities can receive similar grades. Also, grades from different sources cannot be compared meaningfully without detailed knowledge of the learning objectives and grading standards that produced them.

As a motivational device, grading distracts students from the true purpose of their education, which is learning. Students who focus on grades as the goal of their work tend to make choices with grades rather than learning in mind, such as choosing easy courses over challenging ones and expending effort only on what will be tested. They can become anxious, risk-averse, and miserly about learning. These students do still learn, but it is an accidental side-effect rather than a dedicated process. Over time, their misplaced motivation decreases the quality of learning and drains the joy out of the educational process.

These are serious charges. If the community of higher education were convinced by them, we might see dramatic changes in student evaluation. What might these changes look like? I believe that a system of substantive feedback about specific aspects of student work would be far more useful than grades, both to students and to those who wish to compare them.

By substantive feedback, I mean evaluating student work in terms of specific abilities and progress towards specific learning objectives. For students in the classroom, continuous feedback about their learning would place learning where it belongs – in the spotlight – and give them clear guidance for improving. For people like employers, summative evaluations in terms of specific abilities and objectives would provide real information about students. They would also address the rising concerns about grade inflation.

One wonderful consequence of good feedback is that it can help students to revise their work. Allowing students to revise their work removes a major source of anxiety: the fear of undeserved failure. Some students learn more gradually than others, and some are better test-takers than others, so their performance on one-time, high-stress assessments may not reflect their true abilities. Revision gives all students the ability to succeed through hard work. It also allows teachers to give them truly challenging assignments without making them anxious or hostile. The revision process itself is also a highly valuable learning opportunity, since it encourages students to revisit concepts and to analyze their work critically.

Some educational institutions have adopted evaluation systems based on principles like these, but most of the higher education community still uses and depends upon traditional grading. I anticipate needing to assign numeric grades to students for that reason. My goal is to meet that necessity while also providing better information to students. To that end, I plan to use substantive feedback throughout my courses, only converting it to numeric grades at the end.

I should note here that I find it both unfair and unwise to surprise students about their grades. For that reason, I plan to tell students in advance precisely how their final grades will be calculated. For example, I might give feedback using a rubric with course-specific categories, and assign grades by universal criteria: no unaddressed criticism of the work means at least an *A*, and so on. By giving students this information at the beginning of a course, I believe I can make the grading process transparent and fair.

Giving good feedback is certainly time-consuming, and large amounts of it would be infeasible in large classes. However, if necessary I could restrict comments to the few most important aspects and limit student revisions to the few most important assignments. The feedback system could therefore be scaled up or down to meet time constraints, while still providing more substantive information to students than grades do.

Keeping track of students' work and feedback, particularly if they do revisions, sounds overwhelming in practice. However, technology makes it manageable. Electronic work is easily stored, and work on paper – along with written comments – can be scanned for electronic storage as well. I can also require students to submit the original version of their work along with a revision, so that comparing them is easy.

Grading is important in the current higher education system, but it should always take second place to learning. My goal is to meet the existing demand for grades while also overcoming their deficiencies in order to improve student learning. The strategy I have outlined here is intended to strike that balance.

Teaching-as-research in an Engineering Course

1. Introduction

The goal of this project was to improve students' abilities to solve reactor model problems in Environmental Engineering 320, taught by Professor Trina McMahon. Reactor models are simplified versions of complex environmental systems, such as lakes and rivers. They abstract away the details and allow environmental engineers to model systems in standard ways.

Students in the course were having trouble translating word problems into the conceptual framework of reactor models, and then manipulating equations to solve the problems. They could follow and understand a problem solved in class, but their work on tests demonstrated that they had difficulties doing so on their own. There seemed to be a kind of disconnect between understanding and producing.

I worked with Professor McMahon and her post-doctoral student, Angela Kent, to design an interactive web tutorial that guided students through step-by-step solutions of several problem types, and then required them to solve similar practice problems on their own. Its main advantage over classroom demonstrations was that the entire process was interactive for each individual student. The step-by-step problems gave a lot of guidance, but the students had to provide answers from the beginning. Also, the technology was able to point out and explain each student's mistakes right away.

Each problem type had a guided problem and a practice problem, and the reactor model problems also contained graphing exercises that required students to manipulate and answer questions about graphs of their equations.

2. Project Implementation

In Spring 2004, we analyzed midterm exams to determine exactly where in the problem-solving process they were having trouble. We originally expected to find difficulties with graphing and interpreting results, but in fact, more students had problems simply setting up the material balance equation and solving it.

Through Summer and Fall 2004, we designed and created the learning module. It was designed as a Java applet that interacts with a MySQL database, and its basic framework is somewhat separate from its content so that the content can be updated mostly independently of the more complicated programming aspects. Small changes could probably be made by someone who does not know Java, given simple instructions, and we also created web forms to modify the database easily.

In Spring 2005, we used the learning module for the first time. Each of the 50 students in the course received an account in the database, and their answers and mistakes were recorded as they progressed through the module. They could return multiple times to finish problems and to study. We gave deadlines that encouraged them to complete the tutorial in two sections, finishing shortly before the midterm exam.

Most of the challenges we encountered at this stage were technical difficulties and user interface problems. The biggest ones were:

- Students couldn't move on if they were stuck on one problem
- Rounding differences were interpreted as incorrect answers
- Error messages were unhelpful for numerical answers
- Students had some difficulty navigating between pages

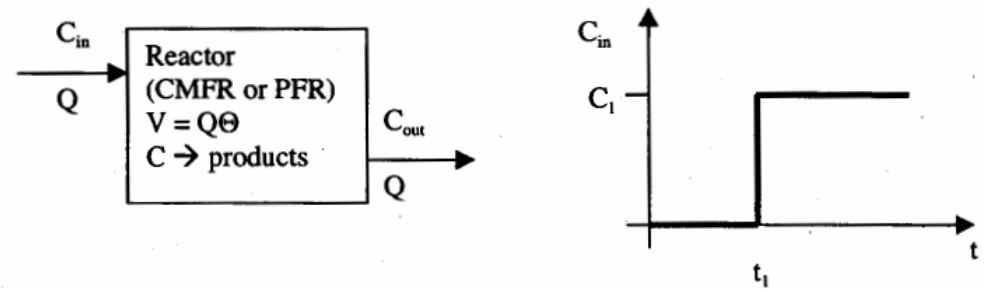
3. Project Evaluation

We analyzed student solutions of a problem in Spring 2004 and Spring 2005 to compare student performance without and with the learning module, respectively. The problem is displayed in Figure 1.

A fluid flows through a reactor with mean residence time given by Θ. The system is illustrated below. The concentration of a contaminant species C in the fluid at the reactor inlet is 0 for all times $t < t_1$. For all times $t > t_1$, the concentration at the inlet is maintained at $C_{in} = C_1$. At $t = 0$, the tracer concentration within the reactor is zero. Within the reactor, the species undergoes first-order decay with rate constant k:

$C \rightarrow$ products $\qquad\qquad$ $R = kC$

For cases (a) and (b), sketch the time-dependent contaminant concentration at the outlet, $C_{out}(t)$, for $t \geq 0$. The concentration and time scales should be properly and completely labeled.



(a) Assume the reactor is an ideal CMFR
(b) Assume the reactor is an ideal PFR

Figure 1: The problem analyzed for student difficulties.

Our analysis method was to classify the types of mistakes students made in their answers into the following six categories:

- misuse of CMFRs (one type of reactor model)
- misuse of PFRs (another type of reactor model)
- mistakes setting up material balance equations
- misunderstanding the characteristic time
- mistakes solving the equations to get an answer
- mistakes graphing and reflecting on the solution

The results of this evaluation are summarized in Figure 2. The largest improvements were in setting up material balance equations and solving them, which were among the worst areas before and which were our primary focus in the tutorial. Clearly we should also add material on characteristic time, which is also a big area of confusion. The fact that students did no better with graphing could simply indicate that the interface problems overwhelmed the intended learning on those pages, or perhaps that they need to draw graphs themselves instead of just manipulating ones that are drawn for them (i.e., our initial approach did too much of the intellectual work for them).

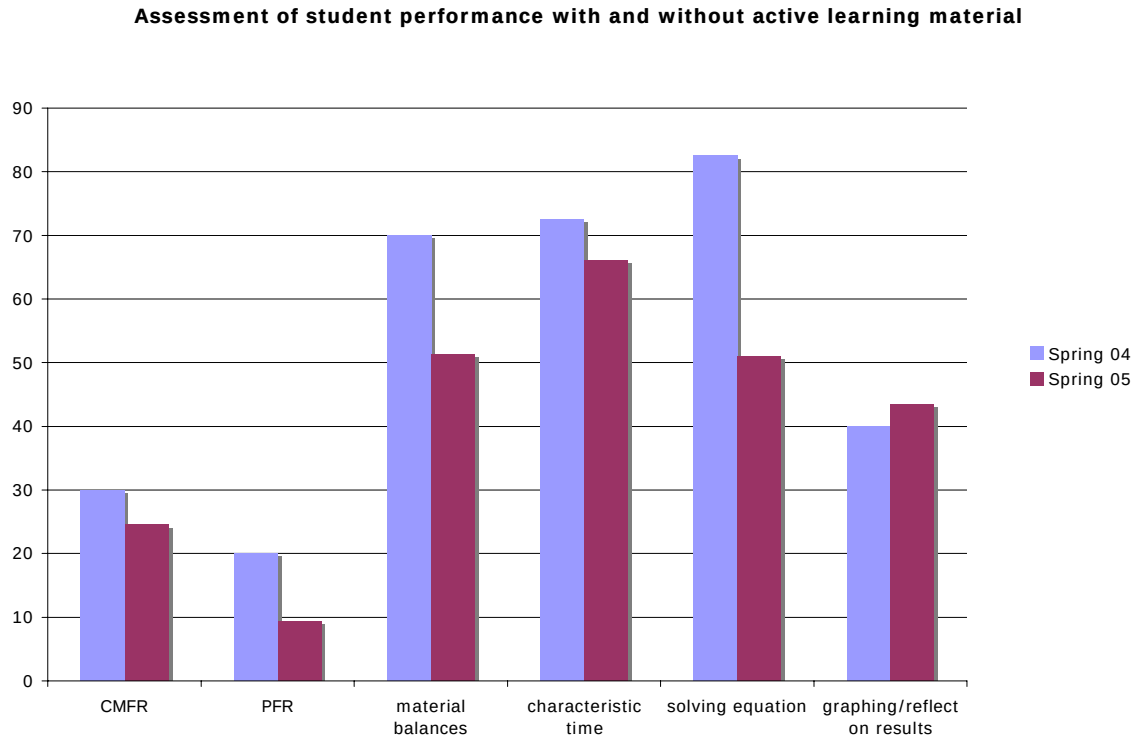


Figure 2: Percent of students struggling with concepts.

We gave a survey to the students after the midterm exam, and in their responses they mentioned the following things as helpful about the learning module (summarized in Figure 3):

- the overview that it gave of the reactor model problem-solving process
- its utility as a study tool
- the practice it gave them
- the step-by-step manner in which it taught them to approach problems
- the way it helped them to set up and solve equations

4. Improvements

We made improvements to the module based on student feedback, and we have begun to use the module again in later semesters. These are the main areas that we addressed:

- Making navigation easier
- Accepting small ranges of numerical answers near the exact answers
- Including a problem that uses multiple reactor models
- Addressing characteristic time
- Improving the graphing problems
- Giving more explanations for incorrect numerical answers

We hope that the improvements will reduce the mistake rate even further in future classes.

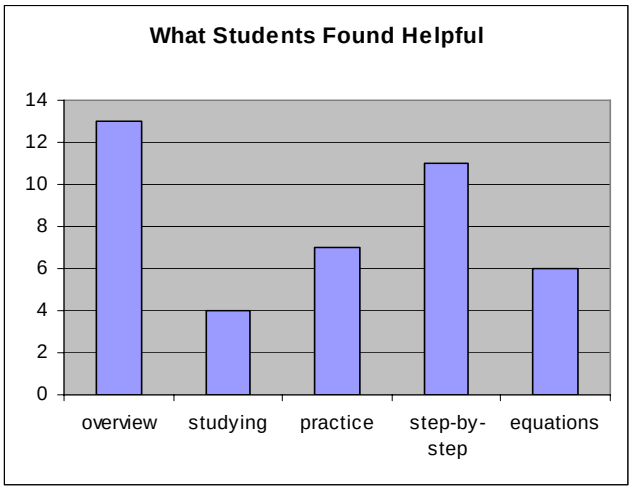


Figure 3: Student feedback.

5. Conclusions

The web tutorial approach does seem to be effective in reducing students' misunderstandings about solving reactor model problems. One lesson that we have learned, fortunately not the hard way, is to make sure to test for student misunderstandings before planning the solution. If we had not done so, we would have designed the module quite differently, and it would not have addressed the actual problems.

Another lesson, which we did learn the hard way, is to pay a lot of attention to user interface issues. When there are interface problems, they become the students' focus, to the detriment of their learning. A good interface will be one that doesn't elicit any comments at all from the students, indicating that they are concentrating on the content.

6. Literature Resources

In problem-solving situations, students find algebraic applications difficult. Many students cannot translate rational word problems into mathematical expressions (Clement, Lewis & Mayer, Lochhead & Mestre, Mayer, Wollman).

Misconceptions during translation from problem tasks to equations are a typical source of error. Students maintain incorrect conceptual systems regarding specific concepts in mathematics. (Leinhardt & Zaslavsky & Stein).

These misconceptions, evidenced by persistent error patterns, have a psychological base. Learners adapt new knowledge to their existing cognitive structures. The problem is that "learners' cognitive structures are difficult to change significantly" (Herscovics).

Addressing these issues requires an approach that considers students' general reasoning processes (English & Warren), accounts for conceptual errors in problem translation (Kaput & Sims-Knight), and includes linked multimedia presentations (Kaput, Mayer & Simes).

Johari demonstrated significant improvement in university students' ability to conceptualize variables and create equations from word problems after exercises incorporating inductive multimedia materials.

References

J. Clement. Algebra word problems solutions: Thought processes underlying a common misconception. *Journal for Research in Mathematics Education* 13(1), 1982.

L. English and E. Warren. General reasoning processes and elementary algebraic understanding: Instructions for initial instruction. *Focus on Learning Problems in Mathematics* 17(4), 1995.

N. Herscovics. Cognitive obstacles encountered in the learning of algebra. *Research issues in the learning and teaching of algebra*. Reston VA, 1989.

A. Johari. Effects of Inductive Multimedia Programs in Mediating Word Problem Translation Misconceptions. *Journal of Instructional Psychology* 30(1), 2003.

J. Kaput. Technology and mathematics. *Handbook of research on mathematics and learning*. New York, 1992.

J. Kaput and J. Sims-Knight. Errors in translations to algebraic equations: Roots and implications. *Focus on Learning Problems in Mathematics* 5(3,4), 1983.

G. Leinhardt, O. Zaslavsky, and M. Stein. Functions, graphs, and graphing: Tasks, learning and teaching. *Review of Educational Research* 60, 1990.

A. Lewis and R. Mayer. Students' misconceptions of relational statements in arithmetic word problems. *Journal of Educational Psychology* 79(4), 1987.

J. Lochhead and J. Mestre. From words to algebra: Mending misconceptions. *The ideas of algebra, K-12.* Reston VA, 1988.

R. Mayer. Memory for algebra story problems. *Journal of Educational Psychology* 74(2), 1982.

W. Wollman. Determining the sources of error in a translation from sentence to equation. *Journal for Research in Mathematics Education* 14(3), 1983.

Teaching-as-research in a Programming Course

1. Introduction

In CS 302, an introductory programming course, students are taught the Java programming language. A Java program consists of several pieces of interacting code grouped into subtasks called *methods*. The main method of the program passes information to other methods, which report their results back.

When I taught CS 302 in 2003 and 2004, I found that students had difficulty with methods on the first midterm exam that the course coordinator administered. Many would misinterpret the task performed by a method, or use one incorrectly. They had the most difficulty writing their own methods—they often gave answers that showed a profound misunderstanding of how methods work.

Most students understood methods much better later in the course, after several large homework assignments that required them to use methods extensively. However, there were always a few students who felt overwhelmed by these assignments; some gave up and dropped out. They needed a learning opportunity that was more difficult than lecture, but less difficult than the large assignments, and they needed it before the first exam.

The goal of my teaching-as-research project was to fill this gap. Specifically, the learning objectives for students who participated in the project were to improve the following skills:

- Reading existing methods and determining the tasks they perform,
- Using existing methods correctly in a program, and
- Designing methods to perform required functions in a program.

These goals correspond to the Analysis, Application, and Synthesis levels of Bloom's Taxonomy. Those are all among the higher levels of the taxonomy, but they also rely on lower-level skills like recall of Java syntax. My project addresses recall in order to support the higher levels of cognition, but concentrates primarily on those higher levels.

I designed an interactive web-based tutorial to give students both theory and practice on Java methods. I chose this technology because it can give students more guidance by giving feedback as they work, but can also require more activity than a lecture or a textbook. Also, I had developed the tutorial system for a recent project and hoped to study its use further.

An additional benefit of this technology is that it can give students selective information according to the specific mistakes they make. This helped me to address the diversity of CS 302: some students start with no programming experience at all, while some start with enough experience to be familiar with many of the concepts we teach. I was able to accommodate several experience levels by presenting background information only when it was needed by an individual student.

2. Project Implementation

In Spring 2005, the CS 302 course consisted of 15 sections with 15-20 students each. I invited the graduate student instructors to have their students use the tutorial (I was not teaching a section that semester myself), and 3 of the sections did. They completed it during a 50-minute class period in a computer lab, with their instructors present to answer questions. The sections that did not use the tutorial either had a normal lecture that day or had an exam review period. All of the sections had covered the same topics in lecture, although small differences between instructors no doubt occurred.

The tutorial reads like a Java textbook, with a textual narrative and illustrative diagrams, except that it asks multiple choice and open-ended questions throughout instead of dictating most information. I have included some images of tutorial pages below, to give examples of content and to show how students interacted with the system.

The page in Figure 1 contains some explanatory text and a multiple-choice question. Students checked boxes to answer the question and pressed a "Check Answers" button. If they answered incorrectly, an explanation appeared and they could try again. Once they answered correctly, the page looked as it does below. Students could still look at the explanations for each choice by holding the mouse over the corresponding box.

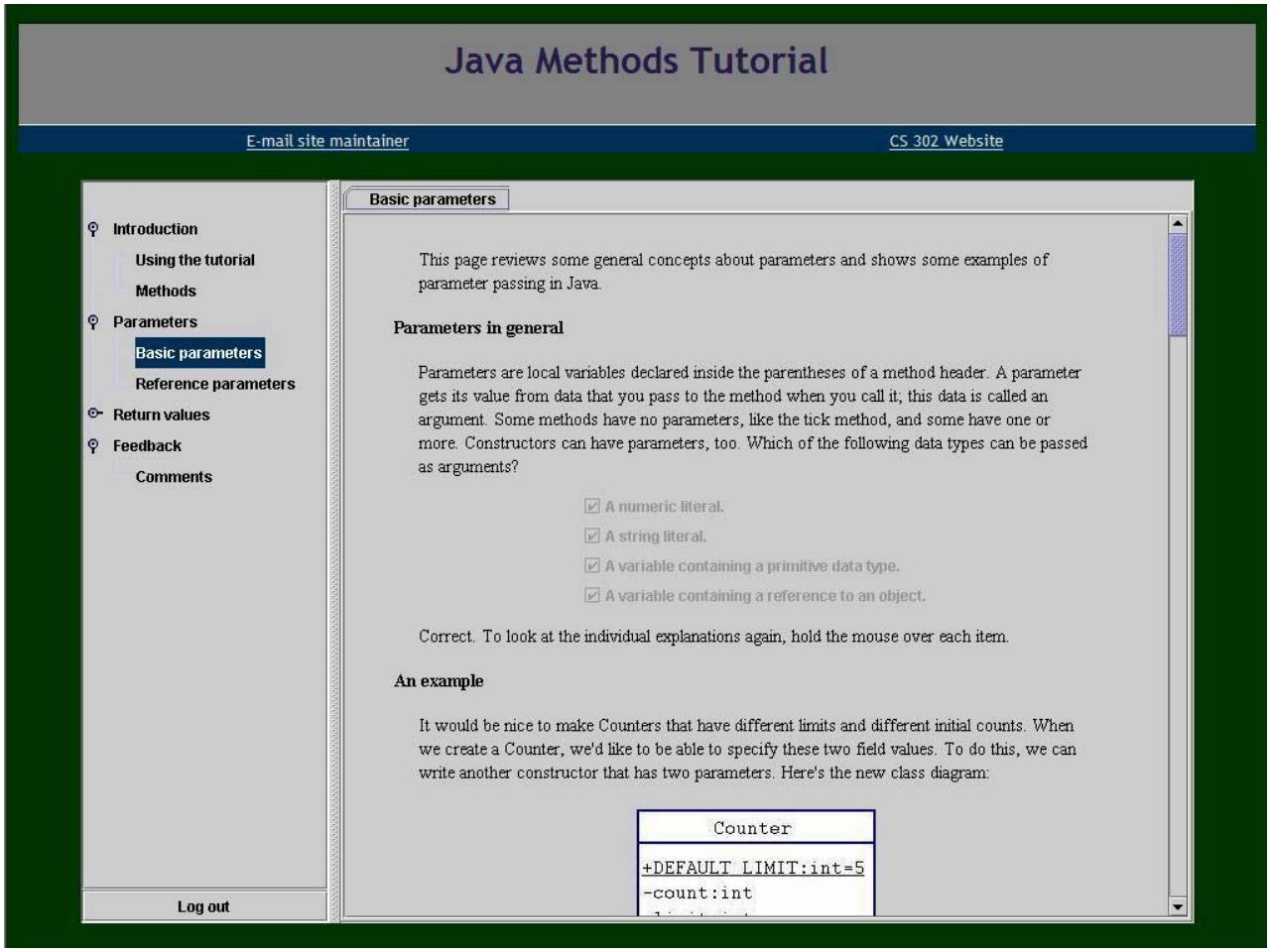


Figure 1: A multiple-choice question

The tutorial's multiple choice questions generally address my first and second learning goals: interpreting methods and using them. Other questions are more open-ended, requiring students to enter code. These generally address my second and third learning goals: using methods and writing them. Students did not receive electronic feedback for these, because the tutorial system (which I designed using Java applets) does not yet have the ability to analyze code. Instead, they were asked to call an instructor over to look over their code.

The page in Figure 2 shows one of these open-ended questions. Most of text in the white box was initially present on this page, but the last three lines were entered by the tutorial user, as prompted by the instructions and the "NEW" comment. One advantage of this format is that it looks like the editor that students use to write programs for homework assignments, so it feels like they are adding to a whole program rather than entering a disconnected answer without any context. The "Save Answers" button allows students to store and change these code entries.

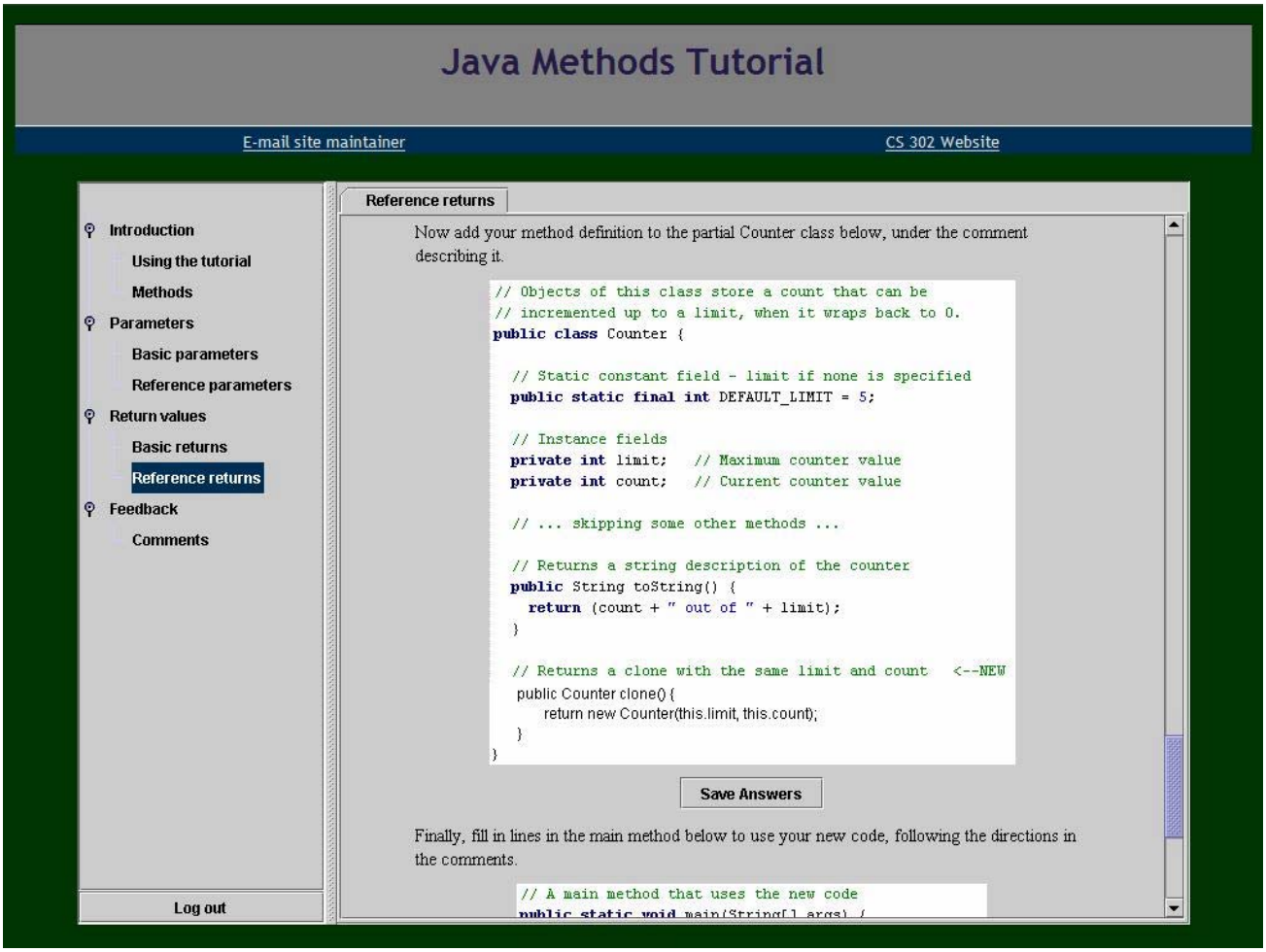


Figure 2: An open-ended question

2. Project Evaluation

Students took their first midterm exam in the evening of the same day that they completed the tutorial. (I would have preferred that they use the tutorial earlier, but it was not ready soon enough.) Because of this timing, it served as more of a review session than as the intermediate instructional material that I intended.

The midterm exam served as an evaluation tool for the project. Some of the questions were related to the tutorial material, and some focused on other aspects of Java. Thanks to the generous help of the course coordinator, I received summaries of students' answers to exam questions. These were anonymous, although they were labeled by section so that I could separate the sections that completed the tutorial from those who did not. By comparing the scores of these sections, I was able to analyze the effectiveness of the tutorial in helping students achieve the learning objectives.

3. Results

Figure 3 reports the average scores of the two categories of students (those who completed the tutorial and those who did not) in three categories of questions (the entire exam, questions related to the tutorial, and questions unrelated to the tutorial).

If we look first at the blue bars, which represent students' total scores on the exam, the results look poor because those who used the tutorial scored 4 points lower on average. However, look next at the control group (no tutorial) and how they did on the two subgroups of questions: they did better on questions *unrelated* to the tutorial material. Then look at this same comparison in the other group of students: they did better on questions *related* to the tutorial.

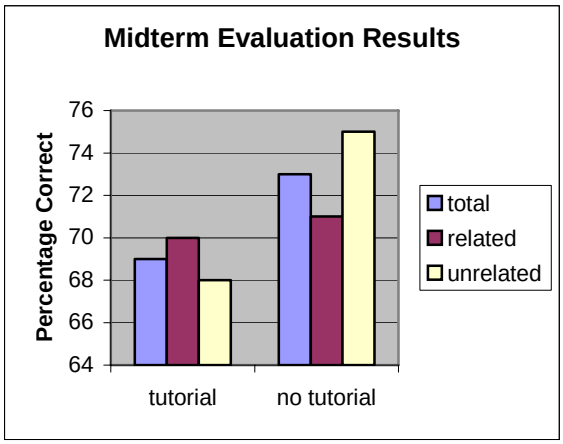


Figure 3: Average exam scores

If we consider students' total performance to be an indicator of their average ability, we will accept that the 3 sections that completed the tutorial had lower average ability. (Perhaps this was why their instructors decided to have them use the tutorial.) Without the tutorial, we would expect those sections to have a similar distribution to the control group, with significantly better performance on questions unrelated to the tutorial. The fact that this relationship was reversed seems to indicate that the tutorial had a positive effect.

Within the exam questions related to the tutorial material, I found three types of questions: those requiring students to interpret methods, those requiring students to use methods, and those requiring recall of Java rules and terms. Figure 4 compares the scores of the two student groups on these question types. The average scores shown are divided by their average scores on unrelated questions, so we can look at this graph as a comparison of performance on tutorial-related questions adjusted for overall ability. The students who completed the tutorial did better on all three types of questions, though the gap is smaller in the "using methods" category, which both groups found difficult.

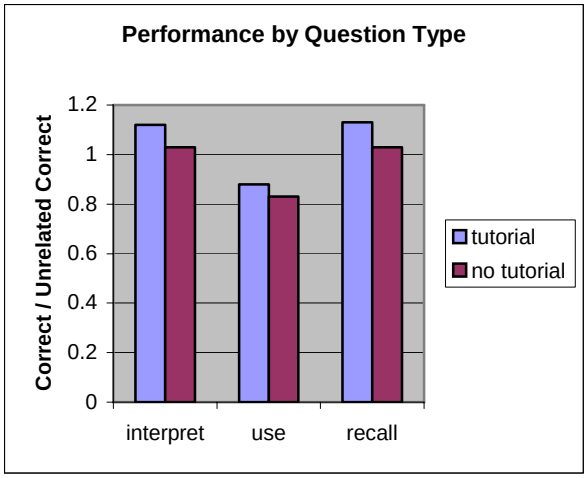


Figure 4: Average performance on related questions

4. Conclusions

There is evidence that students who completed the tutorial did better on tutorial-related questions than they would otherwise have done. The tutorial appears to be helpful for recalling Java rules and terms, for interpreting methods, and for using methods.

Students who responded to my request for feedback said that the tutorial helped as a study tool for the exam, by focusing on what they needed to study and giving them practice. They also gave some suggestions for improving the user interface. For example, one student asked for a way to look back at pages she had finished; there was in fact a way to do so, but it was not obvious enough. I have found in all my experiments with the tutorial system that user interface issues are of high importance, since they can significantly distract students from the actual content.

Another common suggestion was to make the code entry perform automatic analysis of student code. This is technologically complex, but I would like to make the tutorial system capable of doing it. I found that many students did not end up with correct answers to those questions, presumably because they did not ask their instructors to check them. If the tutorial could check them, students would be more likely to correct their errors and therefore their misconceptions.

If I repeated this experiment, I would try to arrange a more even split between sections that completed the tutorial and sections that did not. Most instructors have two sections, so it would be ideal to separate each instructors' two sections between the groups. Second, I would try to arrange the tutorial to be given earlier (perhaps immediately after the relevant lectures).