2013
TEACHING EXCELLENCE AWARD
TEACHING PORTFOLIO

JEFF OFFUTT

COMPUTER SCIENCE / SOFTWARE ENGINEERING

Jeff Offutt

Professor of Software Engineering (MS) (BS)
Mailstop 4A5
Volgenau School of Info Tech & Engineering
George Mason University
Fairfax, VA 22030-4444
Office: 4430 Engineering Building
Fall 2012 office hours: Wednesdays 3:00-4:00

Email: [illegible]
Phone: (1) 703-993-1654
URL: http://cs.gmu.edu/~offutt/
Twitter: JeffOffutt
ORCID: 0000-0002-8657-2557

Students who want to email me should read my suggestions hints for students making requests first.
My thoughts on cheating and plagiarism.
Google scholar   COS bio

(*Names of individual students are elided for privacy*.)

“Once my teacher, always my teacher”

– Chinese proverb

“The one thing they can never take away is an education”

– Unknown

Section 1: Cover Letter

Professors in engineering departments at research schools like Mason have an odd, sometimes contradictory relationship with teaching. I was hired for my research and granted tenure for my research accomplishments. I am recognized in my department for having a prolific publication rate. Where does teaching fit in?

When I interviewed at universities and when I was growing my career, I always downplayed my interest in teaching. I wanted them to think I was a researcher, not a teacher! But I had a secret—I enjoyed teaching and wanted to be good!

One of my proudest accomplishments was being the first faculty member in the Volgenau School to be promoted to Full Professor for excellence in **both** research **and** teaching. When my chair announced that to our faculty meeting, I was relieved to finally come out of the closet and admit that I sincerely like to teach.

Why do I like to teach? Because I am more than a lifetime learner, I am **addicted** to learning. I get bored if I don't learn something new every day. I teach because it gives me an excuse to learn. This same compulsion drives me to teach new subjects and to find new, innovative ways to present knowledge.

I never fully understood that connection until I compiled this portfolio. This reflective process has allowed me to put my 20 years at Mason in perspective, dissect my relationships with my students, and understand how my use of technology has grown and evolved.

I have structured this portfolio in terms of my growth as a teacher, from demanding excellence from my students, to collaborating with them, encouraging them, and finally my current approach of leading them to discover knowledge. Through this journey I have continually adopted new technologies and insisted on using them to improve education, not just make delivery more efficient. I started teaching with chalk boards, moved to white boards with colored markers, then plastic transparencies, and on to power point. I first used the web to disseminate class materials and display examples, then broadcast my lectures to synchronous, simultaneous audiences outside the classroom. I then found I could use the web to extend learning beyond the temporal and spatial constraints of the classroom through interesting and informative discussions, used recorded lectures to run a "flipped" classroom, and finally ran a class asynchronously, completely on the web, while merging groups of students from three different universities on two different continents.

Along the way, my relationship with my students has grown and evolved. Early on, I had a very peer-like relationship, then became more of a big brother, and as I started teaching undergraduate students, developed more of a father-figure approach. Through all of this, one of the most enjoyable parts of teaching has been enduring relationships with people I genuinely like and respect.

I have never put together a teaching portfolio. In fact, I had never seen one until this month. When I reviewed the portfolios of past winners in the CTE offices, I was overwhelmed with the incredible quality of the teachers at Mason, not to mention how much work they'd put into their portfolios.

No matter what happens, though, I already feel like a winner. I asked about a dozen students to write short testimonials, but got 25! They talked with each other, and I started getting letters unsolicited, and former students saying they'd heard and would I mind if they sent in a letter. When I read them I was overwhelmed with emotion. I cried, laughed, and shared with my wife. This is the only award that counts: students telling me that I helped.

I hope you enjoy reading this portfolio as much as I did putting it together.

Jeff Offutt
February 2013

Section 2: Table of Contents

Section 3: Teaching Statement

Jeff Offutt, PhD

Professor of Software Engineering, Computer Science Dept.

Volgenau School of Engineering, George Mason University

Technology should be used to enhance education, not just to cheapen it.

I love everything about teaching. I enjoy organizing the material, lecturing, interacting with students, grading papers, all of it. But my favorite part is when a student is struggling with a problem or a concept, and I can diagnose the difficulty, explain the concept, and see the student get the solution. Seeing that particular light bulb go off, up close and personal, still thrills me. So it may seem counter-intuitive that I have become a strong advocate of using technology to enhance education. However, instead of creating a distance, I've discovered that technology can be used to bring student and teacher closer together. And re-instill a love of learning in both of us! The rest of this document explains this journey, starting with some of my background, my educational role at Mason, and two specific successes I've had with using technology in classes.

My Educational Background

I grew up in an educational desert. My father was the only parent around who graduate from college. Teachers usually taught to the lowest common denominator, and classes were tortuously slow for bright students. Instead of earning respect for learning quickly, I was a target for bullies. Bullying became worse when I skipped fifth grade, and the torture peaked in eighth grade, when the superintendent placed four special-education 15-year olds into a small classroom of 13 year olds. The goal was to let them "graduate grade school" before they quit school. The effect was we learned nothing that year. High school was only marginally better. 40% of my freshman class dropped out, and 10 out of 150 graduates started college. My geometry teacher couldn't do proofs, my chemistry class had no chemicals or even running water, and my senior English class included functionally illiterate students.

By the end of this process, I was poorly educated, but hungry. Starved for knowledge.

My college was what I could afford on my own. It was a small teaching school with no research or graduate programs. Most of the teachers were dedicated teachers, but with limited knowledge and ability. They were there because they couldn't get jobs at better schools. I graduated still hungry.

With luck and high GRE scores, I was accepted into a premier research university for graduate school. My professors were brilliant scientists who ranged from bad to excellent teachers. These two very different university experiences led me to my most enduring principle: *We do not have to sacrifice excellence in teaching to achieve excellence in research*.

I try to exemplify this principle. One of my proudest accomplishments is being the first faculty member in the Volgenau School to be promoted with a rating of Genuine Excellence in both research and teaching. I view research as another form of teaching—where I am both the teacher and the student. My hunger for knowledge has never abated and it has become a passion for teaching. I love to be the teacher I seldom had.

Teaching Principles

This passion has led me for years, and I helped me develop a list of guiding teaching principles.

1. If you don't care about them, they won't care what you say
2. Don't expect them to be students like you were
3. The main correlation with success is frequent detailed feedback—not class size, difficulty, quality of lectures, or the book
4. Respect them as people, even if they don't earn respect from their results

5. Fear is not respect
6. Teach to individuals, but grade anonymously
7. Don’t apologize for doing the right thing, but always apologize for mistakes
8. If you don’t know the answer, help them find who does
9. If they think you want them to learn, they will learn more

These principles are heavily influenced by my kids, and I owe them for helping me become a better teacher.

My Educational Role at Mason

I came to Mason in 1992 primarily because of the Master’s program in Software Engineering (MS-SWE). More broadly, I was attracted to Mason’s emphasis on innovation and its entrepreneurial spirit. I have taught 11 different software engineering classes and in 2003 took on the leadership position as Director of the MS-SWE program. I helped create a concentration in Software Engineering within the PhD in Information Technology program (2000), and an undergraduate concentration in Software Engineering within the Applied Computer Science program (2010).

In 2005 I led a major overhaul of Mason’s MS-SWE program. The MS-SWE degree was created in 1989 and is one of the oldest and most successful in the nation. The changes were both innovative and fundamental. Both students and the industrial advisory board have been overwhelmingly in favor of this curricular modernization.

My greatest joy in leading the MS-SWE program is seeing the students’ beginning-to-end progress. Dozens of our graduates keep in touch through email and Facebook, including the SWE Alumni @ GMU group on Facebook that I encouraged one of my students to create. My latest innovation is a twitter account that I hope to use to keep in touch with our alum. I attend commencement every year so I can shake each SWE graduate’s hand, tell them I’m proud, and wish them luck. My favorite moment was when a 4-year old girl looked up at me with bright tears of joy and pride in her eyes and said “*Do you know what my Mommy did? She **gaditated**!*” I have no idea what that little girl will do in life, but I am 100% certain that someday she too will *gaditate*.

Innovations in Software Engineering Education

I enjoy mentoring students at all levels: undergraduate, MS, and PhD. I was the first in my department to post course materials on my website and one of the earliest adopters of powerpoint in the classroom. I use online discussion boards in every class to expand the conversation. I have created nine courses at GMU, three at each level. Seven had never been taught anywhere and were created without prior models or adequate textbook support. These courses have transformed Mason’s software engineering programs. I created and first taught five of the 13 software engineering courses offered in fall 2012, and the current instructors use my materials. I created the syllabus for two others, turning them over to others to create the content and to teach.

My novel approaches to teaching have had significant impacts on software engineering education at Mason and beyond, and are often copied internationally. Software engineering is by nature interdisciplinary, and my courses incorporate elements from computer science, engineering, management and psychology. I eagerly embrace new teaching methods and have invented innovative techniques that continue to be adopted at universities throughout the world.

In my first year at Mason, I redesigned *SWE 637: Software Testing*, introducing a new book and new material. This changed the emphasis from research to practical engineering that directly supports the needs of our MS students and their companies. Dr. Paul Ammann and I re-invented the course again, this time with our own textbook, *Introduction to Software Testing* (Cambridge Press, 2008). The book is the most widely used software testing text in the world. The high quality slides, example assignments, solution manual, and support software make it easy for faculty elsewhere to teach testing. We donate all royalties to the Software Engineering Scholarship Fund at Mason (over $20,000 so far).

I have been invited to teach software testing at numerous places, including the ARTES summer school in Sweden, Samsung electronics in Korea, Rockwell-Collins in Iowa, Skövde University in Sweden, Ewha University in Sweden, the TAROT summer school in Austria, and the Universidad Politécnica de Madrid this January.

I also created a graduate course in designing and building software user interfaces that was unique in looking at software usability as an engineering discipline (*SWE 632: User Interface Design and Development*). This course is now one of the most popular graduate electives in the Volgenau School. It emphasizes collaboration, critical, analytical, and imaginative thinking in an interdisciplinary way. Elements from this course have been copied at universities throughout North America.

In 1999 I created one of the first courses in the nation on engineering high quality Web software applications (*SWE 642: Web Application Design and Development*). The class was a direct response to the specific needs of the Northern Virginia software industry and has technical depth and immediate practicality. 642 is taught every semester, usually by adjuncts with my materials. No textbooks adequately support this class, so I designed it from "whole cloth." The class is innovative in both content and delivery, and it has been copied by dozens of universities. I have taught this material in China, Sweden, and Austria. One of my favorite uses of technology is in-class case studies, where we look at the UIs of web software, then "open up" the backend to study the details of how the software is designed and built.

I have also created and taught three undergraduate courses: *SWE 205 Software Usability Analysis and Design*, *SWE 432 Design and Implementation of Software for the Web*, and *SWE 437 Software Testing and Maintenance*. SWE 432 is one of the most popular electives for CS majors. SWE 205 is the best loved, with an average student evaluation rating of 4.94. I have also created several research-related courses, including *SWE 763 Software Engineering Experimentation*, *IT 821 OO and Architecture-based Testing*, *IT 824 Analysis of Software for Testing*, and *SWE 825 Special Topics in Web-Based Software*.

Using Technology in Teaching

Three years ago I was very dubious of online classes and even discussion boards. I thank two people, Maricel Medina (an MS-SWE student) and Dr. Sharon Caraballo (of the Volgenau School), for convincing me to open mindedly try new approaches. Their encouragement led me to this essay's leading question: "*How can technology be used to improve education, not just cut costs?*" Quality is always my first priority. The following paragraphs present two classroom experiences that were answered the question affirmatively.

SWE 763: Teaching one class at three universities: This class started with another question: *How can one professor teach the same class at three different universities?*

In 2008, I taught *SWE 763: Software Engineering Experimentation* as a traditional, face-to-face class, one day a week for 2.5 hours. The class featured three weeks of lectures about experimentation, nine weeks of in-class discussions of research papers, and two weeks of students presenting the project results. In 2011, a research collaborator asked me if I could spend a semester teaching SWE 763 at Skövde University in Sweden. That was impossible, but the request led me to the above question ...

Teaching a class at multiple, international, universities presented significant problems. Lectures could not be synchronous. Mason regulations prohibited me from being the "instructor of record" at another university while teaching full time at Mason. Would the students be enrolled at GMU or their home institutions? USA students pay tuition directly to universities but tuition is paid by the government in Sweden. Lastly, whose learning management system (LMS) should I use? It is very difficult to enroll students from one university into another university's LMS, and the poor usability of Mason's bulletin board would not support the kind of dynamic interaction needed.

The solution came in several parts. Mason's Provost gave me special permission to be instructor for the classes in Sweden, allowing students to enroll in separate classes at their home universities. The second part was a happy accident. I got a spam message from a young startup company, *piazza.com*. Their tool is a free discussion board with a modern, social networking style user interface—exactly what I needed!

The course had the same structure as in 2008—three weeks of recorded lectures followed by online discussions. nine weeks of online discussions, and two weeks of student presentations—but entirely online and asynchronous. Once the class was announced, Linköping University (also in Sweden) asked permission for their students to join the class, making us three.

I taught the course in spring 2012 to 20 students, 14 from Mason, five from Linköping, and one from Skövde. I assigned each discussion paper to two students to write summaries and evaluations by Monday evening. Then I used a Swedish custom of assigning a "dissenter" to each paper, who had to disagree with the first two students and post the dissent Tuesday. We spent the rest of the week discussing the papers. Each student also designed an experiment, carried it out, and wrote the results in a conference-style paper. Students selected a project I proposed, or proposed their own ideas. Midway through the semester, they wrote one-page descriptions of their projects, posted them online, and the rest of us offered feedback.

The students quickly melded into one unified class, answering my question affirmatively: *Yes, I could teach the same class at three different universities.* An early impression was that the introductions were more detailed and informative than in a face-to-face class. This foreshadowed the major benefit of the asynchronous format.

The discussions of the research papers were nothing less than outstanding. In 2008 we had a 2.5 hour limit, students often came to class tired, many after a long day at work, and not always prepared to discuss the papers thoroughly. The asynchronous format allowed us to extend the discussions indefinitely, prepare at our convenience, consider other students comments, and add to the discussion anytime. The discussions were both longer and deeper than in the face-to-face class! Unlike in 2008, the 2012 discussions were fascinating and the students taught me as much as I taught them.

The advantages were highlighted during the week of project proposals. In 2008, the students spent five minutes apiece presenting their experimental designs, and then we spent five minutes giving them feedback. In 2012, students posted 10 to 15 minutes worth of material, and received up to four hours' worth of feedback. Some discussions continued for several days.. As a result, their experimental designs were significantly better in 2012. We were truly freed from the "tyranny of the clock"!

The students posted paper drafts on the discussion board, then two classmates read and gave comments. I used PDF's annotation feature to provide comments. At semester's end, students submitted their final papers and presented their results to the class. This was synchronous; students at Mason met in a classroom, and I traveled to Sweden after Mason's finals. I realized then that we had struck gold. In 2008, four of 15 papers (27%) were eventually published. After only six months, five of 20 papers have already been published or accepted for publication (one before the course ended!), one has been submitted for publication, and four are actively being revised for submission. Thus, 25% of the projects have already been published or submitted for publication, and another 25% will be; a potential for half the projects being published!

All students were pleased with the class, and I believe this format can be successfully copied with any discussion class, whether in scientific research or in literature. The discussions and diversity of projects meant the class emphasized divergent thinking, reducing the problem of determining who actually did the work. Moreover, the multi-university format offers several interesting possibilities. If there aren't enough students to support a specialized class, they could join the same class at another university, even on another continent. If a class is taught by someone with unique expertise, students elsewhere could join. This format could scale from 20 to 200 students if professors at other universities participated in the feedback and evaluations. This opens the possibility of what I call "*crowd teaching*," where a diverse team of professors from 5 or 50 universities merge their courses online, all being responsible for part of the content, but each professor being responsible for managing a subset of the students.

SWE 637: Using a flipped classroom: I first heard about "the flipped classroom" in a presentation from the Center for Teaching Excellence. Instead of listening to the professor talk, then doing homework, students in a flipped class listen to recorded lectures at home, then work problems in the classroom.

This semester I flipped my classroom for two weeks in my graduate software testing class. I picked problem-solving material that students often struggle with. I recorded the lectures in 15 minute chunks and posted them online. Students were told to view the lectures before class. In class we worked problems together, then the students started their "homework" and called if they needed help.

The results were quite positive. Five students completed the assignment in class. Much of my time was spent explaining subtle points of the material to students who missed it in the reading and lecture. Other students needed help with some of the basics. It's easy to say "they should already know it," but this format let us fill in holes in their knowledge so they could succeed in this part of the class.

I followed with three questions on our class discussion board: "*Did you view the lectures before class?*", "*Did you feel the flipped classes were useful?*", and "*Do you think you did better on the homework because of the class session?*". All the respondents said yes to all three, and students are asking for more flipping.

The crucial question is whether the flipped model enhanced their l. I compared with the same assignment from the previous two years. The average was 88% in 2010 and 86% in 2011, but 98% in 2012 with the flipped classroom. The lowest score increased from 65% in 2010 and 63% in 2011 to 90% in 2012. Theorizing that a flipped classroom might help struggling students more, I computed the average of the bottom half of each class. These averages went from 75% in 2010 and 74% in 2011 to 94% in 2012!

This experience demonstrates that the flipped model can improve education and also be effective in graduate courses. The flipped model has several advantages: (1) Working problems in a group can help. (2) It's hard to concentrate for an entire class, but students can pause recordings at any time. (3) Flipping lets students go at different speeds, which is a relief to gifted students and a benefit for struggling students. (4) The in-class sessions let professors focus on what each student needs individually, rather than treat all students the same.

What this Award Would Mean to Me

I love engineering, I love computer science, and I love teaching. I love innovating with new topics and new methods of teaching, and I love the diversity that we have at Mason. I believe collaborative learning and divergent thinking are necessary to teaching engineering effectively. A wonderful surprise of these technological innovations is that they allow me to provide **more** individual attention to students. I see that light bulb more often! Using technology allows me to spend more time teaching instead of lecturing.

Ammann and I are working on a second edition of our software testing book. In 2008 we provided powerpoint lectures, which I now think is very 20th century. A 21st century textbook should include complete *recorded lectures*. Instructors should be able to send students to the book website for the lectures, and come to the classroom for clarifications, to work problems, and most importantly, to learn.

Teaching these students at this university is a privilege, made better by the many excellent colleagues I've been fortunate to work with. Of course, any award that recognizes hard work is welcome, especially hard work that is not recognized during promotions and regular evaluations. But I have longer term goals. I want to expand my multi-university course by using crowd-teaching. I hope to take a stronger leadership role in the Volgenau School in teaching excellence in general, and teaching with technology in particular. I want to expand my successes in my own classes, and strongly hope to teach my colleagues the lessons I've learned. **Yes, we can use technology to enhance education, not just to cheapen it.**

Section 4: Teaching CV

Jeff Offutt
Department of Computer Science, George Mason University, Fairfax VA 22030, +1 703-993-1564
www.gmu.edu/~offutt/, offutt@gmu.edu

ACADEMIC EDUCATION

1988	PhD in Computer Science	Georgia Institute of Technology
1985	MS in Computer Science	Georgia Institute of Technology
1982	BS in Math & Data Processing (double major, Cum Laude)	Morehead State University

TEACHING EMPLOYMENT HISTORY

Aug 2005-	*Full Professor* of Software Engineering, George Mason University
Aug 1996-2005	*Associate Professor* of Information and Software Engineering, George Mason University
Aug 1992-July 1996	*Assistant Professor* of Information and Software Engineering, George Mason University
Aug 1988-May 1992	*Assistant Professor* of Computer Science, Clemson University

Courses Taught at Mason

Introduced a large number of innovations in terms of topics, teaching techniques, and style. This includes developing nine completely new courses and substantially revising six others. Also pioneered several innovations for incorporating the web into class material delivery, some of which are now used by many other faculty. My lecture notes for SWE 432, SWE 437, INFS 590, SWE 619, SWE 632, SWE 637, and SWE 642 have been used by several other faculty at Mason and elsewhere.

Course	Title	Times taught
SWE 205	Software Usability Analysis and Design	4†
SWE 432	Design and Implementation of Software for the Web	8†
SWE 437	Software Testing and Maintenance	1†
INFS 590	Program Design and Data Structures	2‡
SWE 619	Software Construction	5‡
SWE 626	Software Project Laboratory	2
SWE 632	User Interface Design and Development	20†
SWE 637	Software Testing	13‡
SWE 642	Software Engineering for the World Wide Web	8†
SWE 763	Software Engineering Experimentation	6†
SWE 825	Special Topics in Web-Based Software	3†
SWE 821	OO and Architecture-based Testing	1†
SWE 824	Analysis of Software for Testing	2†
IT/CS 990	Dissertation Topic Presentation	3‡

† Indicates courses that I created and taught for the first time.
‡ Indicates courses that I substantially or completely revised.
I also taught twelve different courses at Clemson University from 1988-1992 and one class as a PhD student at Georgia Tech.

MENTORING AND ADVISING

PhD Advisees Completed

- Birgitta Lindström, *Testability of Dynamic Real-Time Systems*. Graduated from Skövde University in Sweden, March 2009. (Co-advisor: Dr. Sten Andler.) Assistant Professor, Skövde University.
- Aynur Abdurazik, *Coupling Analysis of Object-oriented Software*, May 2007. Recipient of ITEA Fellowship, 2002. Senior Test Manager, SAIC.
- Supaporn Kansomkeat, *An Analysis Technique to Increase Testability of Class-Component*. Graduated from Chulalongkorn University in Thailand, May 2007. (Co-advisor: Dr. Wanchai Rivepiboon.) Assistant Professor, Songkhla University
- Mats Grindal, *Evaluation of Combination Strategies for Practical Testing*. Graduated from Skövde University in Sweden, March 2007. (Co-advisor: Dr. Sten Andler.) AddQ Consulting.
- Robert Nillson, *Mutation-Based Testing of Real-Time Software*. Graduated from Skövde University in Sweden, October 2006. (Co-advisor: Dr. Sten Andler.) Research Scientist, Google Research.
- Yu-Seung Ma, *Inter-Class Testing Using Mutation*. Graduated from KAIST University in Korea, 2005. (Co-advisor: Dr. Yong-Rae Kwon.) ETRI consulting.
- Roger Alexander, *Testing the Compositional Relationships of Object-oriented Components*, May 2001. Associate Professor, Colorado State University. Schweitzer Engineering Labs.
- Zhenyi Jin, *Software Architecture-based Testing*, November 2000. ITT Industries. GMU CS Distinguished Master's Graduate 1994, Recipient of ITEA Fellowship 1995.
- Li Li, *Object-oriented Change Impact Analysis*. November 1998. Director of Software Engineering, GeoEye.
- Jane Hayes, *Input Validation Testing: A System Level, Early Lifecycle Technique*. September 1998. Tenured Associate Professor, University of Kentucky.
- Roland Untch, *Schema-based Mutation Analysis*, December, 1995. (Co-advised with Dr. M. J. Harrold). Full Professor, Middle Tennessee State University.

Supervisor, Master's Theses

- Chandra Alluri, *Testing Calculation Engines Using Input Space Partitioning and Automation*, MS-SWE, 2008 (Mason)
- Vasileios Papadimitriou, *Automating Bypass Testing for Web Applications*, MS-SWE, 2006 (Mason)
- Aynur Abdurazik, *Specification-based Test Data Generation Using UML*, MS-SWE, 1999 (Mason)
- Ammei Lee, *FGS: A Multi-purpose Laboratory for Software Engineer Research and Education*, MS-CS, 1998 (Mason)
- Eleanor Rizzo, MA, Interdisciplinary Studies, 1998 (Mason)
- Alisa Irvine, *The Effectiveness of Category-partition Testing of Object-oriented Software*, 1994 (Mason)
- Jie Pan, *Using Constraints to Detect Equivalent Mutants*, MS-SWE, 1994 (Mason)
- Christian Zapf, *Distributing Mutation on a Network of Sun Servers*, 1993 (Clemson)
- Tracey Oakes, *A WIMP Interface to Mothra*, 1993 (Clemson)
- David Pressley, *Data Flow Analysis for Generating Statement Coverage Constraints*, 1992 (Clemson)
- Raad Yacu, *An Improved Procedure for Generating Statement Coverage Constraints*, 1991 (Clemson)
- Scott Fichter, *Parallelizing Mutation on a Hypercube*, 1991 (Clemson)
- Stephen D. Lee, *Weak vs. Strong: An Empirical Comparison of Mutation Variants*, 1991 (Clemson)
- W. Michael Craft, *Detecting Equivalent Mutants Using Compiler Optimization Techniques*, 1989 (Clemson)

- Jason Emil Seaman, *Using Symbolic Evaluation to Address the Internal Variable Problem*, 1989 (Clemson)

Current PhD Students

- Lin Deng. Efficient and Scalable Mutation Testing
- Blaine Donley. Web Application Architectures
- Anders Eriksson (Skövde University). Model-Based Testing of Aeronautics Software
- Jing Guan. Testing and Verifying Wireless Network Protocols using Logical Methods
- Nan Li. Software Test Automation
- Upsorn Praphamontripong. Testing Web Applications with Mutation Analysis
- Sunitha Thumalla. Modeling and Analysis of Web Applications

TEACHING HONORS AND AWARDS

- Finalist, Governor's Technology in Education Award, 2012
- GMU SCHEV Outstanding Faculty member, 2009, 2010
- Best Teacher Award, School of IT&E, 2003
- Outstanding Teacher Award, ISE Department, 2003

CURRICULUM DEVELOPMENT

- Co-led team to create an undergraduate concentration in Software Engineering, within the Applied Computing Science degree (approved Fall 2009)
- Led team to create an undergraduate minor in Software Engineering (approved Fall 2006)
- Led team to a major restructure of GMU's MS program in Software Engineering (approved Fall 2005)
- Developed a graduate Certificate in Web Software Engineering
- Led team to develop a PhD Concentration in Software Engineering within GMU's PhD in Information Technology
- With Henry Hamburger of GMU's CS Department, led team to design GMU's PhD in Computer Science, effective Fall 2000
- Participated in major restructuring of Clemson University's MS in Computer Science (wrote first draft of document)
- Participated in major restructuring of Clemson University's PhD in Computer Science (wrote first draft of document)

EDUCATIONAL PUBLICATIONS

- Jeff Offutt and Bill Karabelas, Teaching an Innovative Asynchronous, International, Multi-University Course, in preparation
- Jeff Offutt, Putting the Engineering into Software Engineering Education, IEEE Software, 30(1):96,94-5, January/February 2013
- Paul Ammann and Jeff Offutt, *Introduction to Software Testing*, Cambridge University Press, 2008
- Mahmoud Elish and Jeff Offutt. T*he Adherence of Open Source Java Programmers to Standard Coding Practices*. The 6th IASTED International Conference Software Engineering and Applications. Cambridge, MA, November 2002
- Peter J. Denning, Ravi Athale, Nada Dabbagh, Daniel Menasce, Jeff Offutt, Mark Pullen, Steve Ruth, and Ravi Sandhu. *Designing an IT College*. Seventh World Conference on Computers in Education (WCCE 2001), Copenhagen, Denmark, June 2001

- Paul Ammann and Jeff Offutt. *Maintaining Knowledge Currency in the 21st Century*. 10th Conference on Software Engineering Education and Training, pages 161-172, Virginia Beach, VA, April 1997
- Paul Ammann, Hassan Gomaa, Jeff Offutt, David Rine, and Bo Sanden. *A Five Year Perspective on Software Engineering Graduate Programs at George Mason University*. 7th SEI Conference on Software Engineering Education, Springer-Verlag Lecture Notes in Computer Science Volume 750, Jorge L. Diaz-Herrera (Ed.), pages 473-488, San Antonio, Texas, January 1994
- Jeff Offutt and Roland Untch. *Integrating Research, Reuse, and Integration into Software Engineering Courses*. 1992 SEI Conference on Software Engineering Education, Springer-Verlag Lecture Notes in Computer Science Volume 640, C. Sledge (Ed.), pages 90-98, San Diego, California, October 1992
- Jeff Offutt and C. Funsch. *Lab Manual 1.0 for Computer Science 241: Data Structures*. Clemson University, August 1990
- Jeff Offutt. *Software Testing Technology*. The ITEA Journal of Test and Evaluation, 7(2):18-31, Spring 1986
- Jeff Offutt. *Hints on Writing Style for Usenet*. Monthly electronic posting to the Usenet newsgroup *news.announce.newusers*

EXTERNAL INVITED TALKS AND LECTURES (TEACHING)

- Lecture on *Cutting Edge Research in Engineering of Web Applications* at the 2013 International Summer School on Trends in Computing, Tarragona, Spain, July 2013.
- Two-day invited lecture on mutation testing at University Polytechnique Madrid, January 2013.
- Keynote talk at the Software Engineering Education Conference, Kunpur India, *Using Collaborative Learning and Divergent Thinking to Teach Software Engineering*, February 2012.
- Invited talk at the Innovations in Teaching & Learning Conference, *Teaching an Innovative Asynchronous, International, Multi-University Seminar*, George Mason University, September 2012.
- Invited talk to the Northern Virginia Test Automation Interest Group, *Cost / Benefit Arguments for Automation and Coverage*, August 2011.
- Keynote speaker at the Google Test Automation Conference (GTAC), *Automatically Generating Test Data for Web Applications*, Hyderabad India, October 2010.
- Keynote speaker at SOFTEC, *The Model-Driven Test Design Process*, Kuala Lumpur Malaysia, July 2010.
- Seminar lecture at TAROT, *The Model-Driven Test Design Process*, Austria, June 2010.
- Invited talk at Telechips, *The Model-Driven Test Design Process*, Seoul Korea, October 2009.
- Three day seminar at Sansung Electronics, *Introduction to Software Testing*, Suwon Korea, October 2009, October 2010.
- Full-day seminar at Sogang University, *Introduction to Software Testing*, Seoul Korea, October 2009.
- Invited Lecturer for the ARTES Summer School (A network for Real-Time research and graduate Education in Sweden), August 2002, August 2006, Nässlingen, Sweden

GMU Speaker's Bureau and Community Outreach

I have lectured on *Internet Safety* to numerous groups in Northern Virginia for the GMU Speaker's Bureau. I also spoke twice at Robinson Secondary School in Fairfax, VA, on *Computing and Science*. I regularly volunteer at Oak View Elementary school, most recently teaching a module on binary arithmetic to third graders in the Advanced Academic Placement program. I also coach my son's *Odyssey of the Mind* team, a NASA-sponsored competition that emphasizes divergent thinking, creative problem solving, and collaboration.

EDUCATIONAL SOFTWARE DEVELOPED AND DISTRIBUTED

- *Coverage Web Applications*: As part of the book, *Introduction to Software Testing*, Ammann and Offutt worked with students to develop companion software: a graph coverage web app, a data flow coverage web app, a logic coverage web app, and a DNF logic coverage web app. This software is available on the authors' book web site (*http://www.cs.gmu.edu/~offutt/softwaretest/*) and is widely used by students to check and solve homework, and by instructors to illustrate concepts and grade assignments and exams.

- *muJava*: This project is a testing tool that is distributed under an open source software model. μJava is a mutation testing system for Java programs that supports the object-oriented features of inheritance, polymorphism and dynamic binding. It was built as an international collaborative effort between Offutt at GMU and Yu-Seung Ma at the Korean Advanced Institute for Science and Technology (KAIST), as part of her PhD work. μJava is freely available from Offutt's website (*http://cs.gmu.edu/~offutt/mujava/*) and is currently being used at dozens of universities for research and teaching purposes. We have published three papers about μJava, it was featured at the research tool demo session at the International Conference on Software Engineering in May 2006, and dozens of papers by other researchers have used μJava. Offutt and Ammann use it every year in their graduate software testing courses.

- *IMSCU*: This streamlined mutation testing system was built by several graduate students according to Offutt's specification and under his direction. Two versions were built, one in C and another in Modula-2. IMSCU was used in several course projects at both the graduate and undergraduate level, and as a research vehicle in software metrics experimentation.

Section 5: Reflections and Evidence

Jeff Offutt, PhD

Professor of Software Engineering, Computer Science Dept.

Volgenau School of Engineering, George Mason University

Note: *This section refers to "evidential inserts." Most were printouts that were included in the paper copy of this portfolio, but cannot be included in this soft copy.*

I taught my first college class as a graduate student in 1986. I asked for that opportunity to decide whether I was interested in and suited for an academic job. I taught a third-year survey of software engineering to computer science students. I still remember the book, the syllabus, the semester project, the room, and some of the students. Most importantly, I remember learning that not only was I interested in academia, I **loved teaching**. I love interacting with the students. I love the challenge of finding interesting and illustrative examples. I love the process of organizing the material in ways that made it easier to grasp. I love the challenge of avoiding the mistakes of the worst teachers I'd had, and trying to emulate the best.

Most importantly, I loved the feeling that what I did in the classroom would help the students learn, and eventually be better at their jobs. That's the kind of impact I wanted to have on the world. I'm sure I wasn't a particularly good teacher in 1986, but I wasn't bad either. The students liked me, I liked them, and that helped them learn more.

That class started me on an educational journey that still continues. In the following sub-sections, I discuss how I've grown as a teacher, how my relationship with students has matured, my steady progression through technologies, and the evolution of my educational goals.

5.1 The Demander (*1985-1992*)

When I started teaching in the late 1980s and early 1990s, I was friendly and approachable, but also very demanding. I graded to a very high, exacting standard, and was very strict with rules and my grading criteria. This was common in computer science classes, and still is. I demanded that students perform, and often pushed them hard. At the same time, I was not always clear about my expectations or the grading criteria, which must have frustrated some students. I demanded excellence but did not tell them how to get there. This was also, frankly, the Georgia Tech way, an incredibly demanding school where I obtained my PhD. I carried this attitude into my first faculty position at Clemson University.

5.1.i Technology, Teaching Goals, and Scale of View

In those early days, my technology was the chalkboard, and I remember being excited to move to whiteboard with colored dry erase markers. Color became essential to the examples I worked through for students.

My teaching at this time started by emphasizing computer programming skills, and slowly evolved to teaching knowledge. As much as a pedagogical choice, this reflected my growing maturity as a professional.

When I prepared for class, I took a small view of individual lectures, and I moved from lecture to lecture through a semester, each one distinct. That was my job … I was a lecturer.

I spent my first four years as an academic at Clemson University, teaching graduate and undergraduate computer science courses. Class sizes were capped at 25, but otherwise, with my experience now I realize that I was exploited. The normal teaching load was two classes per semester, but in four years I was asked to teach **ten different** classes. This was a tremendous burden on a young assistant professor, but creating two and a half brand new preps every year also gave me a huge amount of experience. This gave me a basis that has helped me grow as a teacher.

5.2 The Judge (*1992-1996*)

1992 brought a major change in my life and career. After four years as an assistant professor at Clemson University, I realized three things. First, I did not want my daughter (born in 1990) to attend in South Carolina. Second, I had outgrown the department in the sense that I had little left to learn from my senior colleagues. Third, I was becoming a software engineer who thought and behaved differently from traditional computer scientists. I was tired of teaching courses in computer science when I saw so much in software engineering that we weren't teaching. I was tired of swimming upstream in a traditional, conservative, conventional computer science department. Then I found George Mason.

Mason had everything I wanted. Fairfax has one of the best public school systems in the country, which taught girls and boys equally, and had multiple opportunities for intellectually gifted children. The faculty included world-class scientists who I could look up to and emulate. And most surprisingly, it included a forward-looking, innovative, and decidedly unconventional school of Information Technology & Engineering with a department of Information & Software Engineering. The ISE department had a 5-year old MS in Software Engineering, one of only a handful in the country.

Thus I moved to Mason and changed from teaching a mix of computer science classes to teaching only software engineering graduate classes. The many software engineering classes, coupled with an easy path to inventing new classes, was a perfect match to my innovative and creative teaching urges.

One of the first things to change was my relationship with students. At Clemson, I behaved as an older student with the graduate students and a big brother to the undergraduate students. But the average age for MS-SWE students at Mason was 33, two years older than I was! So I related to the students more as a peer—somebody with specialized knowledge they didn't have, but without the kind of work experience they had. This enabled me to learn from them as much as I taught them.

As I developed a repertoire of graduate software engineering courses, I changed my attitude from demanding high performance to being an assessor, judging the quality of their work. I tried harder to define and exemplify excellence, but focusing on judging students work sometimes created unnecessarily adversarial relationships.

5.2.i Technology, Teaching Goals, and Scale of View

My technology in the early 1990s was plastic transparencies printed from powerpoint, the whiteboard, and email. I innovated what I called "electronic office hours" by encouraging students to ask questions through email. The part-time students especially appreciated the flexibility. I also asked the students to give me their email addresses, and I broadcast answers to general questions to the entire class. I was the first in my department, possibly the university, to make such heavy use of email.

In 1994 I discovered the web, and fell in love again. I was the first in the school of IT&E to post my syllabus online, as well as all homework assignments, and eventually my lecture slides. I also created the first departmental website on campus for the (former) Information and Software Engineering department.

Most of our MS students were (and still are) part-time students, working full-time in the software industry. They helped me to make my teaching more practical and less theoretical. I also spent more time teaching them how to use knowledge from the books to become better problem solvers. A favorite class for years, for both students and me, was SWE 632 User Interface Design and Development. The semester project and software user interface evaluation homeworks emphasized analytical writing, creative problem solving, and divergent thinking (words I learned much later as an Odyssey of the Mind coach).

I also evolved the scale of what I was teaching. Instead of looking at individual lectures, I was teaching broader semester-long topics in testing, user interface design, and software design and construction.

Evidential Inserts for Section 5.2

1. SWE 632 Spring 2003: Syllabus, Topics, & Term Paper Assignment

   I taught this class my first year at Mason. It was already on the books, but my directive was "don't teach it like the last guy." I designed a user interface course for engineers. In this initial offering I handed out the syllabus on paper and the students wrote term papers. With over 40 students, most of whom could barely write English, I decided it would be more time-effective to switch to a project in future semesters. I also wanted to make this class interactive and full of discussion, but found it impractical with such a large class.

2. SWE 635 Fall 1993: Syllabus, Schedule, & Readings

   This is a much more technical class and relies on sophisticated discrete mathematical models. My goals in this class were to help students know how to efficiently and effectively test software at all levels, ***and*** *to write better software. That is, by having a deep understanding of what can go wrong with software, they should be better equipped to avoid many of the mistakes. This is a very hard goal to measure, and the strongest feedback was from industrial part-time students who have often told me things like "I thought I would learn to test, but I was shocked to find that I became better programmer."*

3. SWE 619 Fall 1993: Syllabus

   This is an introductory class required by all software engineering graduate students. The syllabus and books I inherited presented the course as a pretty boring, 1980-style of how to design and build small programs. Through several semesters I modernized the course, introducing more object-oriented concepts, and introducing more theoretical bases for the design strategies.

4. ISE website, July 1999

I believe I created the first departmental website at Mason, and it was quickly copied by the CS department and the School of IT&E. This is not the first one I created, but is fairly illustrative of the early days of departmental websites. This is the oldest version I could find on the wayback machine (web.archive.org), from July 1999. The diagram is based on a design modeling language (UML) that is commonly used to design software. Although I meant the design to be both clever and useful, I think it primarily confirms that I made the right career decision by going into engineering instead of marketing.

5.3 The Collaborator (*1995-1999*)

By the late 1990s I had invented several new graduate courses. I completely redesigned our software testing course (SWE 637) to teach practical knowledge for industry use as opposed to the previous research-intensive course. This was more effective for our practical industrial MS students and was a popular elective among MS-SWE and MS-CS students.

I invented a course in software engineering experimentation for our PhD students (SWE 763) that taught students how to design and conduct experiments, and then disseminate the results. After a few weeks of general lectures on research and designing experiments, we spent much of the semester reading and discussing published experimental papers. As a class, we highlighted the positive points of each paper, and identified the negatives.

The way SWE 763 was structured reflected a change in my relationship with students. Instead of telling them what to do, we became collaborators to learn the material together. I treated students in all my classes as professionals and de-emphasized my role as an authority, taking the attitude that we were taking the class together and was more or less the "team lead," a common title in the software industry.

I also renovated an existing software project lab course (SWE 626) that had traditionally had teams of three or four students design and build a moderate-sized computer program. I decided to scale it up, and created one large team of 25 students (a fixed enrollment cap). At the beginning of the semester I served as *customer* and explained to the team what I needed. They then developed system-level requirements, followed by an architectural-level software design. Once that was done, I shifted roles into a *project manager*, and assigned the students into five teams of five, one for each major software component. Each team had to select a team lead who met with me weekly, and an integration specialist who worked with the other teams to ensure the pieces would integrate successfully at the end of the semester. After building the software, they went through unit, integration, and system testing, eventually turning the entire system over to me (functioning as customer again). This multi-team integrated project was unique, and enjoyed by all, although the students did complain about the end-of-semester pressure being a bit too realistic.

5.3.i Technology, Teaching Goals, and Scale of View

My use of technology through the mid to late 1990s slowly evolved, often outstripping Mason's classroom's abilities. I started to use animation in my power point slides to introduce examples one step at a time. For awhile, only a few classrooms had built-in projectors and I often had to lug my personal laptop and a portable projector to class. I took every opportunity to tell the university that "all 21st century classrooms should have a built-in computer and projector, just as all 19th century classrooms had a chalkboard.[1]" I also began to experiment with using the web to show examples in the classroom. Although I found this effective, many classrooms were not yet web-enabled. My next campaign was to claim that a 21st century campus should have universal wifi, a radical idea in the late 1990s.

My educational emphasis was on problem solving. I viewed software engineers as people who spent much of their professional time designing and building software to solve problems.

[1] I am happy to observe that this principle has finally been accepted throughout Mason.

Evidential Inserts for Section 5.3

1. SWE 632 Website Spring 1999: Syllabus, Schedule, Project Description, Evaluation Assignments, Evaluation 1, Evaluation Checklist, and Handouts

   I innovated using the web to distribute syllabi starting in 1998, and quickly migrated all of my materials to my website.

   This was and still is an extremely popular class. My goal was to create a class that was very practical and full of knowledge that students could immediately apply to their jobs in the software industry. Moreover, I wanted a class that was not less work than most other graduate courses in computing, but taught more knowledge. The hundreds of positive notes I've received from graduates (some in the testimonial section) and the hundreds of students who have taken the class indicate those goals were reached.

   I have a new project every year, but with the same structure. The project style, the evaluations, and the evaluation checklist has been copied by many teachers at other universities. The handout page is a collection of resources that I have accumulated through the years, many from students. I started this habit then and still use it now.

- SWE 763 website, Fall 1996: Syllabus, Schedule, & Project
  This class was a research PhD seminar that taught students how to carry out empirical validation in their research. I lectured on experimental techniques, then we spent most of the semester discussing papers from the literature. I can't believe I had to go to the library to find papers, copy them, then create a bound packet for the students. Now they simply Google the papers and print. Or read online.
- SWE 626 Spring 1999: Syllabus & Project Description
  This class was intended to give our students a full project experience in a capstone class. In addition to a major project, I lectured on various aspects of professional behavior. Although a good class, I felt that it had some limited value for our full-time students but almost no value for our part-time students. They knew more about working on a team project than I did!

5.4 The Encourager (*1999-2008*)

The academic years 1999 through 2001 brought another collection of huge changes to my teaching. Looking back, I think this is when I really started learning how to move from competence in teaching to excellence. This is when I learned to teach. And it started with a simple conversation.

I stood in the hall early in Spring 2000, speaking with two other software engineering professors, Drs. Hassan Gomaa and Paul Ammann. One of us commented on a change in the field. Many of our students were starting to get jobs developing software that ran on the web, instead of in traditional software development. And I commented that perhaps we should teach our students something about that … naturally, Hassan turned to me and said "Jeff, you are good at creating new courses. You should teach a special topics class." I had absolutely no idea what I was getting into, but I was getting bored with object-oriented software and agreed to try it in the fall.

5.4.i Web Software Engineering—SWE 642

There was no model for this course. I found a few that were vaguely similar, but none that had the technical depth I wanted or that took an engineering approach. So I had to create a new course without a roadmap or example. That summer I went to a bookstore (ironically, bricks and mortar) to find a book for the course. I was able to locate many badly written skills-based books for practitioners, but not a single textbook! My good fortune was having on speed dial a former PhD student, Michelle Lee, who was managing a team of web software engineers at Verisign. During a 10 minute conversation on my mobile phone on the floor at Borders, she completely changed the fledging direction of my course. She convinced me to go with a new technology (Java 2nd Enterprise Edition, J2EE), and explained to me how traditional software processes, design techniques, user interface, testing, and maintenance did not work with web software. In short, in software engineering as in real life, ***the web changes everything***! Over the subsequent decade, this point has been driven home to me over and over again.

That summer I put together a skill-based design and programming class, using the least bad book I found, and added conceptual knowledge from dozens of resources, integrating them into a collection of lectures. In September I realized we had hit pay dirt. My classroom held 44 students, the roster was full, and I had over 25 students on the wait list! Through the next seven or eight years, SWE 642, Web Software Engineering, had two sections of 40 students almost every semester, and was probably the most popular graduate elective on campus.

Not only was the material unique and different, but the way I taught changed dramatically. At first, I had minimal knowledge and no experience. In Fall of 1999 I wrote a program each week, and then assigned it to the students. I was older and started to relate to students in an avuncular role. I was doing quite a bit of consulting in industry, which allowed me to gain a much better understanding of what my students needed (and didn't need).

In 642, I developed a technique of identifying four or five crucial concepts that I wanted to teach each week. I focused on those concepts tightly in the lecture, and then developed very small example programs that illustrated those concepts. In class, I showed the program running, then we studied the implementation in detail. These examples served as teaching tools in the classroom and examples for the students when doing their homework. Although I considered putting these examples on paper for this portfolio, I do not think this presentation is valuable. I would like to invite the reviewing committee to go to my web server and look at some examples online: *http://cs.gmu.edu/~offutt/jeffwebapps/*. Some of my favorites are *randomString*, which I wrote this month to help me call on students in class randomly; *convert2*, which I use regularly; *check24online*, which I wrote to help my daughter prepare for a fifth grade math competition; *multiply*, which I wrote to help my son practice multiplication; and *QuoteServer*, which I run every day just for fun.

I also developed the idea of an *evolutionary project*. I start by assigning a very simple web application, then each week I ask them to add one small piece of functionality that correspond to the concepts from the lecture. Over the course of the semester, these small steps turn into a larger project and at the end, the students had a modest web application that they can show to their friends. Most importantly, they don't just learn the concepts from the lectures, but *use* the concepts in a working program. They were then ready to then apply these concepts in effective ways to real projects at work.

Over the years I have been flattered to see my course copied at dozens of universities. Some instructors contact me and others don't. I post most of my materials on the web and see people using my lecture slides, the examples I invented and built, and my homework assignments and projects. Imitation is definitely the most sincere form of flattery!

Evidential Inserts for Section 5.4.i

- SWE 699 Syllabus & Schedule (Fall 2000)
  I barely knew what I was doing, but we had fun learning together. Over the years since, students have come into the class knowing more and more, so I have had to adapt the material. The technology keeps changing, which also causes the course to constantly change. A big problem at first was the lack of infrastructure for students to run their programs: we didn't have a server or a system administrator to run it. At first I did all of this on our departmental server, spending up to 20 hours a week, then started training my TAs. Eventually I was able to convince the Volgenau school to buy a server and hire a system administrator who understood web applications. Now this infrastructure is used by dozens of classes at Mason and it is a model for how to support this type of education.
- SWE 642 Syllabus & Schedule (Spring 2009)
  By 2009 the course had matured dramatically. We had two sections every semester of 40 or more students. It was a unique class offering unique knowledge that was not available at more than a handful of universities in the nation. But the job market in this area was and still is huge.
- SWE 642 Homework Assignments (Spring 2009)
  That semester I introduced an "AP" project that was modeled after high school AP courses. It allowed students with more background to implement a project that was more interesting. Several students chose this option and enjoyed it much more than they would have enjoyed the normal homeworks, and learned more. I now spend a lot of effort reaching out to our intellectually gifted students.

5.4.ii Usability and Web Software—SWE 432

Another major change at the beginning of the new decade was that I returned to undergraduates after 10 years in a graduate only department. In Spring 2000 the Systems Engineering department asked us to create a course about usability on the web for their students. Since this intersected two subjects I had been teaching, I volunteered. Again, there was no model and this class was the first of its kind in the country, if not the world.

The class was designed to take a systems view of web applications, focusing on the user experience and concepts such as the client-server programming model and maintenance, with only a modest amount of time spent on programming. Much to my surprise, when I got the roster the first week of class, I had one systems engineering student, 39 computer science students, and another 15 computer science students on the wait list! I immediately started changing the course to be more about design and building web applications to meet these students' needs.

I have now taught this course eight times, and other teachers have taught it four or five more. It is by far the most popular senior elective course for computer science majors (ironically since it's not really a computer science course). The philosophy of teaching usability for the front end and design on the back end that supports the user interface allows me to emphasize a principle goal of engineering, creating **high quality** products.

This course is popular because it's fun, challenging, and interesting. And of course, it's widely known as a jobs course—the software industry has a huge shortage of web programmers. Even in the worst part of the recent recession, software companies were desperate to hire well educated web software engineers.

SWE 432 also became a strong recruitment tool for our MS-SWE program. Immediately after we started offering it, more undergraduate CS students started applying to the software engineering program. The course taught them the difference between computer science and software engineering.

Evidential Inserts for Section 5.4.ii

- Letters from students in support of the class
  - **XXX**, BS-Math 2013
  - **XXX**, BS-ACS/SWE 2012
  - **XXX**, BS-CS 2013
  - **XXX**, BS-ACS/SWE 2013
  - **XXX**, BS-IT 2013
- SWE 432 Syllabus, Schedule, Homeworks (Fall 2001)
  The topics included material from two graduate courses, SWE 632, User Interface Design and Development, and SWE 642, Web Software Engineering. The homework assignments built on top of each other to result in a small web application. This is the hybrid homework / project model that I called an "evolutionary project" in section 5.4.i.
- SWE 432 homeworks (Fall 2007)
  By the 2007, I had taught the course six times and it had matured. I recommended students work with a partner to encourage collaborative learning and about half did.

5.4.iii A Software Engineering Minor and Software Testing and Maintenance—SWE 437

After years of talking, the software engineering group decided to jump fully into the undergraduate end of the pool. This was partly motivated by the continuing separation of software engineering from computer science, and partly by the merger of my department, Information and Software Engineering, with the former Computer Science department. In our "new home," we were expected to regularly teach undergraduate students and we wanted to teach courses that students wanted and that we believed in.

We created a minor in Software Engineering in Fall 2007, introducing new courses in Software Testing and Maintenance (SWE 437) and Software Architectures (SWE 443). I designed SWE 437 and taught it for the first time in Spring 2008 before turning it over to Drs. Richard Carver and Paul Ammann.

SWE 437 was designed to be heavily interactive, with time each week given to in-class examples and group-work. This is very rare in computing classes and enthusiastically received by the students. They particularly liked my physical examples to illustrate software concepts, like the marble ball drop toy that I used to illustrate maintenance and unit testing.

Evidential Inserts for Section 5.4.iii

- SWE 437 Syllabus, Schedule (2008)
  I had dreamed of an undergraduate course that combined software testing and maintenance for years, and was glad to finally have the chance to propose it as part of our minor.
  Although I designed the course and taught it for the first time, I am responsible for so many other courses that I have not taught SWE 437 since. We scheduled it for Spring 2009 and 18 students signed up, but we could not find a teacher for SWE 642, which had 44 students, so the chair elected to cancel SWE 432 and move me to SWE 642. I never imagined a university could cancel a class with 18 students enrolled! And yes, many were disappointed. Paul Ammann has taught it since then, and introduced some very interesting material such as JUnit and Test Driven Development.
  I was surprised by the lack of readings about software maintenance, so once again I was stuck with creating material and lectures from poorly written research literature.
- SWE 437 Homework 2, Stutter for the web (2008)
  A common project nowadays is to "web enable a legacy application." Since the original software was designed for a different environment, this never succeeds. I describe it as putting a suit on a street bum—his clothes may look pretty but he still stinks. (Patriot Web is a web enabled legacy system, which is why it is so bad.) I took an old program, modified it to make it worse, and asked the students to perform the maintenance task of webifying it. The assignment made them understand the value of good engineering practices—comments, logical design, code structure, etc

5.4.iv Textbook Author

I never planned to write a textbook. I occasionally thought about it, but would always reconsider, assuming it would impact my research productivity. But in April 2003, Paul Ammann came into my office, saying "we had a problem." The problem was that he was scheduled to teach the testing course (SWE 637) in the fall, but the textbook was out of print and unavailable. My unkind response was "no, **you** have a problem." We searched about a dozen books from my book shelf, quickly disqualifying each for being too narrow, too badly written, too theoretical, not theoretical enough, or just full of incorrect statements. None were written for classroom use, but targeted the larger market of professional software developers.

He went away discouraged and I went back to writing my paper. But Paul came back the next day with a solution to our problem. "No, Paul," I said, "this is your problem, not mine." (To be fair, it was also my problem because I was scheduled to teach the course in the spring, but I didn't want to face it.) Surprisingly, Paul's solution that we write our own book! I initially said "*no, hell no*!", but Paul eventually wore me down.

We spent much of that summer designing the book's organization. Our pedagogical approach was markedly different from previous books. Instead of organizing the book along the software development process (how to do unit testing, then class testing, then integration testing, then system testing), we identified then exploited fundamental theoretical commonalities among all these levels to collapse well over 100 testing techniques into about 20. We also chose how much theory to put in and wrestled with how to define sloppily developed techniques from the research literature. We finished the summer with drafts of two or three chapters, which Paul used that fall.

Our initial goal was to have a good quality textbook for our class, so we didn't approach a publisher until we were almost finished. We also did not set deadlines; simply deciding that we would "write more" for the students each semester. We were strongly motivated by needing a book to avoid canceling our favorite class.

Our colleague at Simula Research Labs in Norway, Lionel Briand, summarized the result best. "*At first I was disappointed that the book was so short. Then I thought it looked too simple. Finally, I realized Ammann and Offutt had applied genius: The book covered everything I knew and more, and was short and simple because of the uniquely creative organization.*"

In 2005 we did a market survey and found only about 30 software testing courses in North America, two thirds of which were research seminars. Our course was at the Masters level for professional software engineers. But in the five years we were writing the book, we started getting inquiries from colleagues elsewhere. "I saw your book on your course website, when will it be finished?" "Can I use drafts of your chapters in the class I'm teaching?"

As we came close to a full draft, our goal broadened. We realized we weren't writing a book just for ourselves, but for the field of software engineering. Then we discussed our frustrations with other books, and decided we didn't just want to provide a book to our colleagues. We wanted to offer a complete course package so that a teacher who wanted to offer a testing course, but did not have a strong background in testing, could use our materials to teach a good course.

The book was published in 2008. That fall it was used by more than a dozen teachers around the country. Most of these classes didn't even exist in 2005! Our publisher tells us the book now has about 80% of the worldwide classroom market in software testing. It has been translated to Chinese, and is available on BitTorrent[2]. Our biggest problem now is that we need to find time to update it for a second edition. The book allowed me to expand my scope of teaching far beyond Mason classrooms. Despite my initial reluctance, I can now say that the only thing I'm prouder of is my children.

Evidential Inserts for Section 5.4.iv

- Book cover art

  Paul and I had fun with the cover. The publisher gave us a small budget, and we worked with a professional artist to design a unique and memorable cover picture. The tornado above the tester's head identifies technologies used to build software, which are being integrated mentally to test software. The tester is supposed to be gender and race neutral (the artist insisted on blonde hair to match the color scheme). This book is known in the software testing community as "the tornado book" and "the green book."

  URL to cover: http://www.cs.gmu.edu/~offutt/softwaretest/frontcover.jpg

- Book website

  We took our goal of providing an entire course, not just a book, seriously. We maintain the book website (http://www.cs.gmu.edu/~offutt/softwaretest/). It offers high quality power point slides, and solution manuals for students (partial) and the instructor. Putting these on the web make them "living documents" so we can update them when mistakes are found. A novel and unique service is that we include support software that helps students check homework, lets teachers run examples in class, and grade students' homework problems, all through the web.

- Donation page

  Both Paul and I were sensitive to the criticism of forcing students to buy books that "lined our pockets." So we decided to forego all royalties for the book. They are donated to a Software Engineering Scholarship Fund, which has primarily been used to send software engineering graduate students in our department to research conferences. We were recently told that the fund is approaching the $25,000 mark, which will make it an "endowed scholarship."

  http://www.cs.gmu.edu/~offutt/softwaretest/fund/royalty.html

[2] BitTorrent is used to share copyrighted material for free. Although bothered at first, a friend (Dr. Ron Ritchey) convinced me to stop worrying by pointing out "Your book is worth stealing! How many other software engineering textbooks are on BitTorrent?"

5.4.v Leadership of the MS-SWE Program

In 2004 I became Director of the Masters in Software Engineering program. This added new dimensions to my educational duties. I was responsible for scheduling the software engineering courses (an amazingly inefficient process at Mason, I might add). Since the software engineering group is stretched very thin, this includes finding adjunct professors to teach up to half of our courses. Unlike other fields, software engineering adjunct faculty have full time jobs in the local software industry and donate[3] an evening a week to come teach. Not surprisingly, managing nine or ten adjuncts is quite time consuming. I also became the main point of contact for students, responsible for graduate admissions, and evaluating graduation applications.

I looked at my role as a leadership position, which led to a broader view of my role as an educator. Instead of teaching lectures or courses, I was responsible for the entire program. At the same time, my portfolio of courses had grown so that I had primary responsibility for teaching five popular courses, despite my teaching load of four courses per year. I started turning over my courses to other faculty, including adjuncts. They used my syllabus, my slides, my examples, and often my homework assignments and projects to teach my courses.

In 2005 I led a major revision to the MS in software engineering program, the first since the program's creation in 1989. The major reason for the change was to modernize the program to reflect changes in the software industry. We reduced the number of required courses from six to four (of 10 total). This gave students more flexibility by allowing them to choose more electives. We also introduced a new required course called *Distributed Software Engineering* to reflect changes in industrial practice. Although I designed the course and set the syllabus, I have not taught it. We also introduced four emphasis areas to guide students through the diversity of courses we offer.

This established a theme that I've continued as MS-SWE director. We constantly update the curriculum and the classes to keep up with this fast-changing area.

I also always attend convocation. Our students worked hard, sometimes for years, and completing a Master's degree is a huge step. If I am not on stage hooding a PhD student, I serve as Marshall to help the graduates find their seats and the stage. I make a point to shake every software engineering graduate's hand and tell them all how proud I am. Many students from the CS department's other MS programs (CS, ISA, INFS) also take my courses, and I seek them out as well. I especially enjoy meeting the family, children, spouses, parents and grandparents.

Let me describe two of my favorite graduation memories. I was congratulating one woman after the ceremony, and a 4-year old girl stepped in between us. She looked up and said "*Do you know what my Mommy did? She graditated!*" That little girl was the proudest person in the Patriot Center! I don't know what else she will do, but I'm 100% certain she will someday "graditate" herself.

[3] They get paid, so "donate" is not entirely accurate. But they all have full time jobs, most with high salaries, so the compensation Mason provides is barely pocket change.

Another was when I congratulated one of our outstanding MS graduates with a 3.9 GPA. His wife took my hand and thanked me very sincerely. It turned out that he took his last class with me, SWE 632, User Interface Design and Development, and got his very first B in his life. She thanked me for teaching him humility! I was appalled that I broke his lifetime 4.0 GPA, and expected him to be angry or bitter. But quite the opposite, he said he really was thankful. His mother and his wife had been telling him for years that understanding people is important, and my course finally made him believe it.

More recently I have started to use social media to support software engineering at Mason. A former student and I started the SWE Alumni @ GMU facebook group, and last fall I started twittering about software engineering (#GMUSWE).

Evidential Inserts for Section 5.4.vi

- Letters from students describing my leadership attitude toward teaching
  - **XXX**, BS-CS, MS-SWE 2010
  - **XXX**, PhD 2010.

    XXX *was a PhD student at Skövde and I co-advised her through a research collaboration with Dr. Sten Andler*

  - **XXX**, current PhD advisee

    XXX *was struggling with her PhD advisor last year, and I agreed to accept her as my student. "Rescue" might a more appropriate term.*

  - **XXX**, MS-SWE 2009
  - **XXX**, BS-ACS/SWE 2012, current MS-SWE

    XXX *is an intellectually gifted student who never had opportunities in public school to learn faster or learn more (**acceleration** and **enrichment**), as students in Fairfax County do. I have worked hard to reach out to* **XXX** *to "draw him in" to the community of scholars and "draw him out" to reach his potential. He's doing well and I'm extremely proud of him.*

5.4.vi Technology, Teaching Goals, and Scale of View

During these years I grew older, again changing my relationship with my students. Instead of collaborating them in a peer-like fashion, I developed more of an avuncular relationship. They were on a path that I had already traversed, and I had the experience (and gray hair!) to help them find the way.

A very important difference in this period was that I learned to **encourage** instead of demand or judge. At the first meeting, I tell students what I expect by laying out their responsibilities and balance that with my responsibilities. I call out the students I've had before and welcome them back. I tell them that most of them can get an A, and clearly explain how. When smart students fall behind, I approach them individually, sometimes after class, sometimes with a written note, and sometimes through email. I often check GPAs, and if a student is performing sub-par, I ask why. When weaker students struggle, I tell them I know the material is hard, but if they try hard enough they will get most of it. When a student misses a lot of classes, I tell him that he will do better if he comes. A big change was in my grading. Instead of marking what's wrong, I mark what should be there, and train my TAs to do the same. I think I learned this from being a parent … be positive, not negative, and clarify expectations. I now tell students regularly how important the book is, and how helpful it is to read ahead. The knowledge is in the book, and I'm simply a guide helping them prioritize its contents and understand the complicated parts.

At the start of my teaching career I read a paper that claimed students do better with frequent and clear feedback. I committed myself to returning assignments and exams promptly and with substantial feedback. In the 20 years since, I am very proud that I have returned **all** assignments and exams back the **very next class period** in every class I have ever taught. My colleague Paul Ammann had an idea a few back along these same lines. He replaced the traditional class-long midterm exam with weekly 10-minute quizzes. I now give quizzes in almost every class and go over the answers immediately. I have found many advantages: students are encouraged to keep up with the class, they get immediate feedback on how well they understood the material, attendance is higher, and overall grades have gone up. This was a great innovation that I give Paul a lot of credit for.

I became better with power point presentations, and studied books on how to connect with audiences. I think this made my lectures more enjoyable, clearer, and better organized. I found that eye contact, smiles, and simply walking around does wonders to keep students engaged. I decided the in-class conversation was often not enough, so I started using Blackboard's discussion board. However it was a struggle. Students clearly did not want to join discussions on Blackboard. It took me awhile to learn why … a point I return to in section 5.5.

My broad educational goals started to evolve as well. Instead of focusing on problem solving, I viewed my job as teaching students to **think** better. This became a theme in my classes, and when I have time, I toss in specific lectures on topics such as the scientific process, experimentation, time management, and effective work habits.

Instead of teaching subjects such as testing and user interface design, I began to view myself as preparing professionals. All software engineering classes are focused towards teaching professionals how to develop better quality software, and although each course has a different emphasis, they fit together to teach the students to think with a quality mind-set. I now make many references to other classes within my lectures to help students think deeper about how the subjects fit together.

Evidential Inserts for Section 5.4.vi

- Responsibilities

  I share these with the students in most of my classes on the first day.

Responsibilities of Professor

- *Prepare useful and interesting knowledge for you*
- *Post materials on class website before class*
- *Come to class on time, prepared to teach*
- *Offer challenging but reasonable homeworks and tests*
- *Grade fairly without bias*
- *Return graded work promptly with educational comments*
- *Goals:*
    - *Have interesting lectures*
    - *Make the class fun*
    - *Use technology appropriately*

Responsibilities of Students

- *Come to class on time*
- *If you miss a class, learn material on your own*
- *Never miss the first meeting of any class!*
- *Listen to all instructions*
- *Turn in assignments on time*
- *Ask for help when you're confused*
- *Read the material*
- *If you disagree with my policies, disagree politely*
- *Goals:*
    - *Read before class*
    - *Learn enough to earn a good grade*

- A weekly quiz from SWE 432, Fall 2012

  This is the key version. I go over the answers immediately after the quizzes and ask for a show of hands of students who think they got full credit. We have a quiz every week in the first 10 minutes of class.

5.5 The Leader (*2009-2013*)

My most recent evolutionary step joined several threads that had been gathering steam for years. The software engineering group expanded our presence in the undergraduate area by creating a Software Engineering Concentration within the Applied Computer Science degree. The ACS-SWE program requires students to take many computer science classes, and several software engineering courses. With our resource limitations (only four software engineering faculty!), we are not yet in a position to create a complete major. Nevertheless, this has dramatically changed my role and my perspective as a teacher.

Another thread was more personal—I was learning how to teach from my children. As my daughters, four years apart, matured into high school and then began college, they changed my view of education. I was able to understand my students in a different way. I spoke at my older daughter's Theory of Knowledge class (in Robinson High) and wound up bringing several concepts back to my classes at Mason. I got to see some interesting innovations at Thomas Jefferson and adapted those to my classes. For example, I learned that collaborative learning, which is usually strongly discouraged in computer science, is particularly beneficial for teenage girls. Both daughters took leadership positions in their high school activities, and I started reading up on leadership principles and passing the advice on.

After 20 years in my research field and publishing almost 150 refereed papers, I was also assuming more of a professional leadership role. In 2007 I took over as editor-in-chief of the leading journal in software testing, Software Testing, Verification, and Reliability. In 2008, I led a small team to establish a new major conference, the IEEE Conference on Software Testing, Verification, and Validation. Now in its sixth year, ICST is the largest, most successful, and most influential of all IEEE software engineering are conferences.

Finally, I was becoming more interested in the next steps in using technology to deliver high quality education. I was tiring of the lecture model, but had not been satisfied with the benefit of Blackboard's discussion board, and was suspicious of the distance education model that was used by colleagues in my department. The most common is simultaneous / synchronous, with faculty lecturing to a live audience in the classroom while students listen to the lecture and watched the slides through the internet. Students complained about technological glitches and instructors ignoring the online students. I owe a great deal of thanks to Maricel Medina-Mora, a graduate student in my program, and Sharon Carabella, our former associate dean of undergraduate affairs. Maricel encouraged me to do more with technology, and gave many specific suggestions. Sharon argued that education could be online asynchronous without loss of quality, and convinced me by inviting me as an online guest into her class.

These four things have instigated a revolution in how I teach over the last four years. Instead of a demander, a judge, a collaborator, or an encourager, I now view teaching as a kind of leadership. A teacher is a leader, and yes, good leadership involves all of the above—demanding effort from the followers, judging their results, collaborating with them, and encouraging them to succeed. These changes have changed every aspect of my teaching—my lecture style, my use of technology, my relationships with my students, and my view of the goals of teaching.

Evidential Inserts for Section 5.5

- Leadership rules, originally authored for my daughters

  This list of rules started somewhat whimsically, playing off of Gibbs' rules in NCIS. But I wound up capturing ideas that have helped me as much as my daughters. As a direct result, I went on to write the Teaching Principles, highlighted in section 5.5.4.

- Proposal for the ACS-SWE program

  This was a joint proposal with Drs. Paul Ammann, Hassan Gomaa, and Sam Malek. Paul did most of the work putting this together.

- Letters from students on my leadership
  - **XXX**, PhD 2007
  - **XXX**, BS-CS, MS-SWE (MS thesis advisee)
  - **XXX**, BS-ACS/SWE 2012
  - **XXX**, BS-ACS/SWE 2012

Leadership Rules

A leader is best when people barely know he exists, when his work is done, his aim fulfilled, they will say: we did it ourselves.

— Sun Tzu

Leadership is hard. Too many people confuse it with authority and power. And too many in positions of authority are not good leaders. A good leader doesn't need authority to lead.

I compiled the following list over a period of years. They have come from everywhere ... Sun Tzu, Reader's Digest, leaders I have followed, my own experience.

Most of these are unordered, but #1 is definitely #1 to me as a leader, teacher, and parent. Note that this list can't be complete.

1. If you don't care about them, they won't care what you say
2. They won't do more than you do
3. A leader is not powerful, but responsible
4. A true leader works for all he or she leads, not the other way around
5. Don't take success to your head or failure to your heart
6. To be a good leader, you must first learn to follow
7. A leader is proactive, a follower is reactive
8. Their time is more valuable than yours
9. Only throw temper tantrums on purpose
10. Get to the point!
11. Tell the truth, especially if the news is bad
12. Don't apologize for doing the right thing
13. Always apologize when you do the wrong thing
14. Say thank you
15. If you don't know the answer, know who does
16. Leadership is service
17. Good leaders are good under pressure
18. A leader has to be trusted
19. Seek first to understand, then be understood
20. Leaders think about the future, followers the present, complainers the past

"*Try not to become a person of success but rather to become a person of value*" (Einstein)

"Be the chief but never the lord" (Sun Tzu)

5.5.i Software Usability Analysis and Design—SWE 205

The ACS/SWE proposal included a new class on usability. We wanted to teach students to focus on usable software early in their education, before they developed bad habits. Thus we came up with a concept of a lower-division course that taught usability **without** requiring programming or other software engineering skills. The only prerequisite is English 101 and the course is accessible to almost any university student.

Once again, I was responsible for creating a new course "whole cloth," without an example or guide. But this time I had a very clear vision of exactly what I wanted—educational goals, results, learning objectives, and the overall "feeling" during the course. I wanted it to be accessible and fun. I wanted students to leave with a different view of the importance of software usability, and with habits that would last a lifetime. In short, I wanted to fundamentally change the way they think. And I was confident that I knew the material deeply enough. What I wasn't confident enough was (1) whether I could find adequate readings and (2) whether I could involve the students in an active learning style. This is rare in computing disciplines!

My search started with books and given my previous experience inventing classes, I wasn't optimistic. I vaguely remembered a book from my youth, *The Design of Everyday Things*, by Don Norman, first published in 1988. Happily, it was still in print! Even better, when I started reading, the examples seemed a bit quaint but the concepts still applied. Norman's book allows me to discuss usability in terms of common physical items like doors and water faucets, then shift the concepts to software. The next book I found was *Don't Make Me Think: A Common Sense Approach to Web Usability*, by Steve Krug. It was modern, software specific, but did not require knowledge of how the software is built. And the title is definitely my all-time favorite! It puts a fundamental principle front and center: a technological artifact engineered for humans should be usable by humans without having to think. What we **want** to do should intuitively and immediately match **how** the UI lets us do it. Thankfully, both books are easy to read, easy to understand, and fun.

My next question was how to get students involved. Software engineering students tend to be fairly introverted and quiet. Many are downright shy. But I had a secret weapon; my 20-year old, quiet and shy civil engineering student daughter. On a 4.5 hour drive to Blacksburg, she told me how to breathe life into my new class. I asked her how can I get engineering students to participate, and she gave me incredibly useful advice.

1. Require participation as part of the grade.
2. Give the students clear guidelines for how much participation they need.
3. Use technology: Some people have a lot to say but have trouble in the classroom. Some are shy and some just don't think fast enough, but will have interesting thoughts later.
4. Give them lots of support whenever they talk. Especially the shy ones. And very especially the girls.[4]

[4] My daughter is accustomed to being one of 7 or 8 women in a 50-person class. CS classes at Mason are even more extreme, with about 5% female students. SWE classes usually have 15% to 20% women, which we think reflects our inclusiveness and focus on collaborative work.

5. Never never never criticize a comment in public, even if someone says something stupid.
6. Make fun of yourself and maybe students you know well, but not students you barely met.
7. Convince them you're a great father.
8. Show that you know their culture, but don't try to imitate it. Make fun of yourself for being a bit old fashioned, but don't make fun of them for being cutting edge.
9. Don't dominate or control the discussion. Get it started and prod, but try to stay out as much as possible so the students feel it's their discussion.
10. Use Legos. Everybody loves a good Lego example. Or any other physical prop. (It took me four years to think of a Lego example; I'm sure it's easier in Civil Engineering.)

This was great advice. I handle #1 and #2 by making participation 25% of the grade, and I using a formula for counting participation. I knew #3 was going to be problematic and was for two years.

I deal with #4, #5, and #6 by reminding myself every day that I'm an encourager and a leader, not a demander or a judge. It makes teaching more fun and the students respond positively. If I tease, I try to tease gently, and if I'm not sure about the student I stop him or her after class and ask if it's bothersome. Most students don't mind as long as they feel I'm joking, not criticizing. For example, this semester I have a marketing student in my class, which is a bit unusual. When I first called role I mentioned her major, and told her that a lot of marketing is done through the UI so I could see a connection. Later I told the class not to use the term "user friendly," as it doesn't carry enough meaning to be clear in engineering. "Except for Alicia, she can say 'user friendly' because it's a good marketing term." After class I asked her if she felt offended and promised not to make another comment like that if she doesn't like it. She just laughed and told me she was thinking exactly the same thing.

I approach #7 and #8 by talking about my own kids, their friends, and emphasize that we actually like each other. Twice my daughter has called me while I was in class. I just put her on speaker and told her to say hi to my students. I'm now a father figure, and the important point I'm trying to make is that if I'm one of their friends' fathers, I'm the semi-cool one that they don't mind meeting, as opposed to the parent they want to avoid. If they like me, they will learn more.

#10 is fun. I talk about usability of my power point clicker, my phone, my watch, and my car. We take mini-field trips to look at doors and walkways on campus. I brought in my son's baseball bag to point out its usable and unusable properties. Next week I will take in his new thermos, which has a spoon cleverly stowed in the lid.

As I said, #3 was a challenge. I taught SWE 205 first in Spring 2010 and used Blackboard's discussion forum. Sadly, Blackboard, and the discussion forum in particular, violate almost every single usability principle in class. Students hate using it, and dislike it more as their knowledge of usability grows. As a result, the online discussion was less than I hoped for. Then in the summer of 2011 I found Piazza[5], a free, web-based forum for class discussions. Its strength is its usability. It's difficult to explain the details here, but the overall effect is

[5] http://www.piazza.com

quite different. Instead of feeling like they come to the professor's party at the professor's house, where they will never quite feel comfortable, students feel like they're having a party at their house and the professor might drop in from time to time. It was perfect for my needs, and in the two semesters I've used Piazza in this class, online participation has been more than twice what it was before. In fact, most students now accumulate more participation points than they need, and some accumulate many more.

I summarize the course learning goals by writing three things on the board the first day of class:

1. January: Patriot Web sux
2. March: Patriot Web is difficult to learn, slow to use, and has an excessively high rate of user errors.
3. May: I can design a better Patriot Web

I use Patriot Web as an example because the students know it and dislike it, but don't know precisely why. My strategy is to engage them by letting them vent about bad software (everybody loves to complain, right?), then assign engineering principles to the problems, then talk about specific ways to improve the design. Much of the course material is principle-based and I illustrate the principles with real-life examples.

I also tell the students that everybody in the room can get an A if they try. That's not true in all my classes and I wouldn't say it if I didn't believe it. But it is true in 205. So far, I haven't gotten an entire class to try hard enough to earn all As, but I still hope.

I turn about a third of the class meetings over to the students. At the beginning they present their homeworks to the class. Last week the assignment was to find one real-life usability problem (*UP)*, explain the problem and how it affects them, and how to improve it. Most posted on Piazza, and they took turns in class presenting their UPs. I get tired of going in alphabetical order, so an innovation this semester is a name randomizer. With my TA's help, I wrote a small web program to accept a sequence of strings (student names), and randomly choose one. The educational benefit was that it gave **ownership** of the class to the students. After each student finished, he or she chose the next presenter by clicking on the Submit button. My role was to sit in the back, nod and smile encouragingly, and when students omit something, I ask.

I implemented many innovative teaching ideas in SWE 205. The online discussions have been terrific. For example, on the first homework this semester, students have been busy commenting on other's homeworks. Why? Because it's fun! And most importantly, they're learning. The example and discussion days grow through the semester. I ask students if they have any UPs to present, and ask them to bring in examples of good and bad user interfaces. Sometimes we take mini-field trips so I can show them real-life physical examples. For example, we look at a safety / usability feature in the stairs in Innovation Hall, and an entrance door to the Engineering Building that has a sign, where I point out the principle that "*if an interface needs a user manual, something is wrong.*" Last year several students suggested better designs of that door. Another student once invited the class to the parking lot to better explain a usability issue with his car. We left the classroom 15 minutes early, and most of the class trekked out to the parking lot.

When I first designed SWE 205, I didn't know the term *active learning*, but now I realize that's a key aspect of the course. I also didn't know the terms *divergent thinking* and *creative problem solving*, but have been using them in SWE 632, SWE 642, SWE 432, and SWE 205 for years. I learned them coaching my son's Odyssey of the Mind team. Many of my projects feature divergent thinking and creative problem solving because I intentionally leave the requirements open-ended and give assignments that have many different solutions.

A surprising advantage of divergent thinking is that every student comes up with a different solution, thus making plagiarism much harder. Or at least, easier to detect.

At this point in my teaching career, I can say that SWE 205 is my all-time favorite class. It requires a minimum of work with the maximum learning benefit. It's fun to teach and fun to take. I'm teaching it for the fourth time and we have had a good feeling in the class every semester. And more students take it every semester. It is required for ACS/SWE students, but more than half the class take it as a free elective. I enjoy the diversity of having marketing, psychology, electrical engineering, and English majors in the room. The workload is low but the learning is high, meaning we're maximizing return on investment. Last year two students attended the class without enrolling because they heard good things about it. This semester a graduate student is attending because he heard about SWE 205 when taking SWE 432, and wants to learn.

It's my favorite, but it may not be my coolest class. I give that honor to SWE 763 …

Evidential Inserts for Section 5.5.i

- Comment on the class from one of our undergraduate ACS/SWE students, **XXX**
- SWE 205 Syllabus & Schedule, Spring 2013
- SWE 205 homework assignment 1, Spring 2013
  This assignment helps the students start to connect the notion of usability with everyday frustrations in life. This motivates the rest of the material.
- String Randomizer web application for SWE 205
  This helps the students take ownership of the examples and discussion days. It's also fun. In Spring 2013, we have seven women in a class of 24 and the program selected six in a row, which led to great mirth in the classroom. It was a great ice breaker! If I could figure out how to make the program do that the next time, I would.
  Randomizer web app: http://cs.gmu.edu:8080/offutt/servlet/randomString
- SWE 205 homework assignment 3, Spring 2012

  This assignment takes an important concept, mental model, and asks the students to connect it to their real life. Some struggle to make that connection, but they all understand after this assignment.

- SWE 205 homework assignment 4, Spring 2012
  This is an analysis and writing assignment. After this assignment I always comment on the depth of analysis (which varies) and point out some common grammatical mistakes. (One of my pet peeves is using an apostrophe to pluralize an acronym.)
- SWE 205 interface evaluation checklist

I initially designed this checklist for SWE 632 more than 10 years, and adapted it for SWE 205. It effectively guides students to thinking about what's important.
Checklist: http://www.cs.gmu.edu/~offutt/classes/205/assigns/checklist.html

- SWE 205 homework assignment 6, Spring 2012
  Okay, what college student won't think it's cool to get credit for playing a game? And following my best advice, I have "MMORPG" right in the handout, but I always mangle it in class to show that I know about "their culture," but I'm not really fluent. The students don't realize they're learning until later. This is all about leadership rather than classic teaching.
- SWE 205 homework assignment 7, Spring 2012
  This week I talk about analysis in terms of stating assertions, building assessments, and finally making decisions. The assignment is to apply this in a small team situation. This is one of the students' favorite meetings. Last year one student concluded by saying "now I know I chose the wrong major and I'm going to change."
- SWE 432 homework assignment 11
  *This is an assignment from SWE 432, but I put it here because it emphasizes divergent thinking and creative problem solving. This is late in the semester and students have been building an interesting project in small steps. In this assignment I suggest four possible extensions and let students pick three. I don't say how, and I explicitly tell them: "*You can format the screens as you wish as long as you follow the usability guidelines from class. You can also add additional functionality. Get creative and have fun!" A few of t*he best students implemented all four functions, and added more that I hadn't thought of.*

5.5.ii Software Engineering Experimentation—SWE 763

I first taught SWE 763 in the spring of 1996. The research field of software engineering had been expecting more in terms of experimentation and many of our students were not keeping up. I structured the course with a semester-long project and three distinct sections. I spent three or four weeks lecturing on topics involving experimentation in general and specific to software engineering. Then we spent most of the semester reading and discussing experimental papers from the literature. Each week one of us would present a paper, then lead the discussion until it tailed off. We discussed two or three papers per week. They also submitted summaries of half of the papers (their choice).

I presented a long list of possible topics and each student chose one or proposed something related to their research interests. Through the semester they designed an experiment, carried it out, and wrote a paper. We took one week in the middle for the students to present their experimental designs and let the rest of us critique them. I also spent a week teaching them about writing and giving research presentations. In the third section of the course the students presented results from their experiments, conference style.

I taught the class like this four times and had great success with the students. Several of the projects went on to become their PhD dissertations, and almost a quarter of the papers were eventually published. I considered this a very successful class!

But then in the spring of 2011 I got an interesting request. I have an ongoing research relationship with Skövde University in Sweden and one of the professors there, Birgitta Lindström, asked me to come teach SWE 763 there. I couldn't possibly get away for that length of time so I declined, but I kept thinking … why should an ocean be such a barrier? It was about time to teach this course again at Mason. Could I leverage teaching it here to reach students in Sweden? The synchronous / simultaneous model that I had been experimenting with would make the time difference (six hours) a problem. It would also mean the Swedish students would effectively be second class citizens. So I started wondering if this class could be taught asynchronously. At first, this seemed counter-intuitive because the entire class was built around in-class participation. But what if we typed on the computer instead of talking in person?

So I broached the idea with Birgitta and she was both dubious and intrigued. And we started talking about practicalities. Could their students get onto our Blackboard? That turned out to be really difficult. Could our students get onto their Blackboard? That would be easier, but still awkward. Would their students pay Mason tuition? Nope. Students don't even pay tuition in Sweden, and certainly won't pay us! So I started wondering if we could use something on the web for our communication. That's when I got a crucial spam!

Actually, my department chair got spammed, and forwarded it to us. A new startup company had launched an online educational software application earlier that year and was looking for users. They had a model of students asking questions online, which could be answered by other students or the professor. It was free, open and web accessible. We can also format text with HTML and Latex. The user interface was exceptionally well designed. The founder of the company was a former Facebook employee and had obviously learned a great deal about the usability of social networking applications. It seemed an ideal tool to help one professor teach the same class to three different universities on different continents!

So I tried Piazza instead of Blackboard during fall of 2011. I quickly found out that its user interface was so engaging that students loved to sign on. Discussion went up by at least an order of magnitude!

I then found out that a professor at Mason is not allowed to be "teacher of record" at another university while simultaneously teaching a course at Mason. So I asked for an exception from Provost Stearns. He was gracious, even enthusiastic in his response:

On 11/29/2011 9:55 AM, Peter Stearns wrote:

> *absolutely, really interesting initiative. We do some team teaching with technology, particularly with Russian institutions, but this carries further and I look forward to the results. Best wishes.*

With Piazza and permission in hand, I started planning in detail. I also got a request from another school in Sweden, Linköping University. Two, three, why not? The more the merrier.

The structure of the class was the same. Except instead of delivering the lectures in the classroom, I recorded them with Camtasia and posted them online. The students viewed the lectures and posted comments. For section two of the class, I assigned two students per paper to write a summary and a critique, to be posted online by Monday (5:00pm EST, 11:00pm euro time). Then I borrowed a Swedish idea and assigned a *dissenter* to each paper who was to disagree with the paper and the two summaries, and post by Tuesday. The rest of the students then joined in the discussion online.

I was absolutely stunned by the results! In 2008 we had 2.5 hours per week to discuss two or three papers. In 2012, based on a word to time conversion formula I found, we spent four to five hours discussing the papers every week. And in my opinion, the discussions were deeper and more thoughtful. Why? Being freed from the clock meant we could go as far as we wanted. And instead of arriving in class after a long day of work, tired and stressed, students could join the discussion when they felt fresh. And instead of losing a thought that came on the way home, or the next day, they could come back and add to the discussion. And all 19 students participated fully, including two students who I knew to be very quiet in the classroom.

The week we discussed the project proposals was when I realized this format wasn't just a convenience, but it actually improved education. In 2008, the students had 12 minutes apiece, seven to present their project and five to get feedback. Online, the feedback went on and on, and in some cases kept coming. One student literally got hours of feedback (mostly from classmates) and by the time we finished had turned a lousy project proposal into credible. He also learned about presentation and experimentation.

The results were fantastic. All papers were interesting. Four of 15 papers from 2008 were published in refereed conferences and journals (26%). The spring 2012 class completed in May, so this comparison is preliminary. Of 19 papers, three have already been published, two have been submitted for publication, two are actively being revised for submission, and students are gathering additional data to revise and submit two others. Thus, 29% of the projects have already been published or submitted, and another 21% will be; a potential for almost half the projects being published!

This class was transformative for me as a person and an educator. I have always believed that learning should be fun, but this was the most fun I've ever had, and I learned as much or more than I've ever learned in a class as student or teacher. I started with a simple goal: teach the same class in three different places at the same time. I hoped to solve the puzzle with technology that did not reduce the quality of the class. What I discovered was a formula that technology to greatly **improve** the quality. I am confident the students learned far more than in a traditional classroom setting. Equally surprising was that my workload actually decreased. I didn't have to spend 2.5 hours per week talking or pushing faltering discussions. The students taught each other a lot that I used to have to teach myself. The only negative was one particularly outgoing student who said she missed hanging out with her classmates physically. But she liked the class enough to write a positive letter, the first in the inserts for this sub-section.

I already received a number of accolades for this class. I was a finalist for the Governor's Technology Award (documented in an insert). The Piazza company, a startup of about a dozen people, have called me out multiple times, including writing a blog about the class. I was invited by Provost Stearns to be on a panel at Mason's Future of Higher Education Forum last fall.

I hope to repeat this class in the future. Indeed, both universities in Sweden have asked me. And I would like to generalize the lessons learned. This was a PhD seminar course with highly motivated students. A key aspect was scale: by having students from three universities we were able to reach a critical mass that turbo-charged the discussions. Would it work as well with 20 students at each university? I believe this format could scale from 20 to 200 students if professors at other universities participated in the feedback and evaluations. This opens the possibility of what I call "*crowd teaching*," where a diverse team of professors from 5 or 50 universities merge their courses online, all being responsible for part of the content, but each professor being responsible for managing a subset of the students.

How many other classes could succeed with a similar formula? Which aspects would work in other classes, which would need to be adapted, and which were unique to this class? I hope to answer these questions by trying a similar approach in different courses. I also want to tell more people about this experience and am writing a paper, *Teaching an Innovative Asynchronous, International, Multi-University Course*, which describes the course.

Evidential Inserts for Section 5.5.ii

- Letters from students in support of the class
  - **XXX**, PhD student
  - **XXX**, visiting PhD student from University of São Paulo, Brazil
  - **XXX**, MS software engineering student
- SWE 763 website, Spring 2012: Syllabus, Schedule, Papers & Project
- Piazza blog. How to Teach One Class in Three Places: Innovator of the Week: Jeff Offutt
  I met the founder of Piazza when she visited Mason, and she was so excited about my class that she asked one of her staff members, Phil Soffer, to write a blog about it.
  Blog: http://blog.piazza.com/2012/05/10/how-to-teach-one-class-in-three-places/
- Application to the Governor's Technology Award
  I was nominated for this award and named a finalist, but came in second. The winners were a group of high school teachers who brought robotics into the classroom. I would have voted for them too!
- Piazza.com

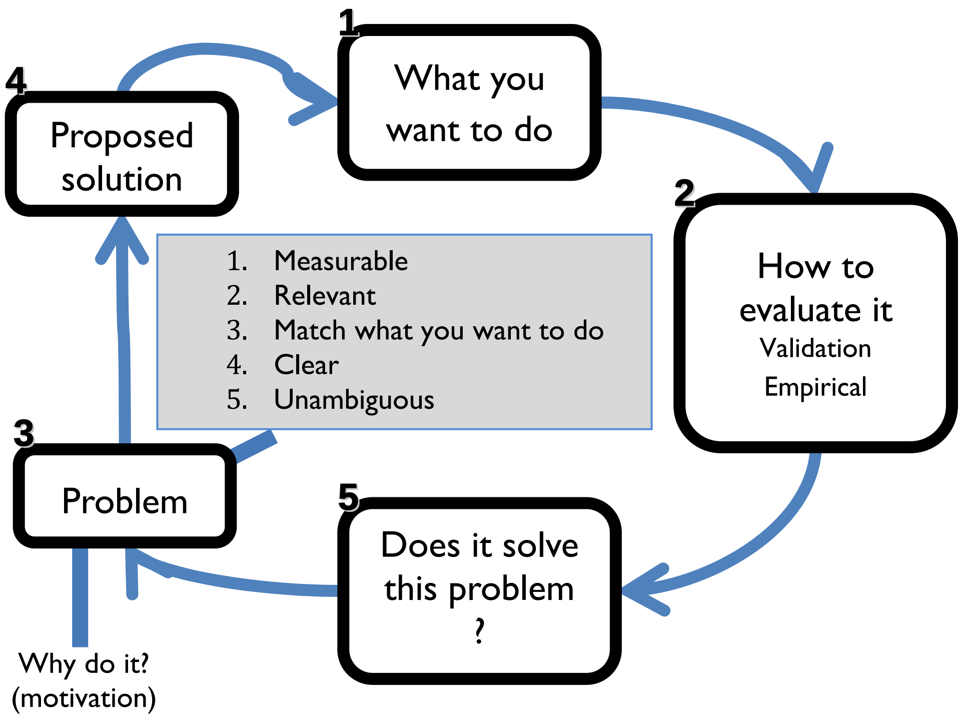
This is a unique class and the printouts are not sufficient to truly see how it works. I invite the committee to investigate the class online through the above URL. Once logged in, I suggest several postings below. You can access each through the URL below, or by modifying the trailing number in the URL in your browser:

- *Student introductions:*
  *https://piazza.com/class#spring2012/swe763/***9**
  *https://piazza.com/class#spring2012/swe763/***12**
- *An announcement and instructions*
  *https://piazza.com/class#spring2012/swe763/***38**
- *Lectures and discussion*
  *https://piazza.com/class#spring2012/swe763/***3**
- *Experimental design discussion (12 minutes total in 2008)*
  *https://piazza.com/class#spring2012/swe763/***186**
- *Summary and dissenters*
  *https://piazza.com/class#spring2012/swe763/***220**
  The students took full advantage of the formatting capabilities and included information that I never expected. By this time, they had taken charge of the class and my leadership role was to sit back and monitor.
- *Students submitted paper drafts by uploading them to Piazza.*
  *https://piazza.com/class#spring2012/swe763/***244**

5.5.iii Technology, Teaching Goals, and Scale of View

Until writing this section, I had not realized how much personal growth as a teacher I've achieved in the past four and a half years. For the first time in years, I am truly balanced in teaching at all collegiate levels: lower division undergraduate, upper division undergraduate, masters, and PhD. In undergraduate courses I look at my students as I do my children, and understand better the difficulties they face and how hard they are trying. I like them more, and think they like me. I function as a senior manager to MS students and as a mentor to PhD students. Whereas at the start of my career I viewed my job as lecturing and grading, I now lead my students through semester-long journeys of discovery.

In fall 2011 I was helping a student outline her PhD proposal. She was having a common problem—separating **what** she wanted to do from **why** she should do it. That is, she wasn't sure how to define the problem. I eventually drew the following diagram:



The numbers are important. Most engineering students start with box 1, knowing what they want to do, and usually know how they want to evaluate. Getting to the problem (box 3) is quite difficult. This diagram allowed me to emphasize the connections, that it's okay to start with the **what**, but there has to be a problem (**why**) and a solution that leads to the what. Then the hardest part is often connecting the validation to the problem … did the research really solve the problem?

Armed with this presentation, I offered to teach IT 990, Dissertation Topic Presentation. It is a one hour course that all PhD students should take as they prepare their proposals. I am teaching it now for the third semester and the students are very appreciative of these insights. We meet twice at the beginning of the semester to define a proposal. Mid-way through we meet again for students to present their problems, thesis statements, and validation plans. We have an online asynchronous week where the students view lectures I recorded on writing and presenting, then they present their proposals in two meetings at the end of the semester. I teach 990 as an overload, which means voluntarily in addition to my normal teaching load.

In the past few years I have learned how to take our learning beyond the temporal and spatial confines of a classroom in multiple ways. I have active and interesting discussion boards in every class. Students enjoy the opportunity to discuss class-related ideas outside of the classroom, and learn from each other. Indeed, I learn from them as well. I also learned to record lectures and share them online to support an asynchronous online course (SWE 763, in section 5.5.ii).

Last fall in SWE 673, Software Testing, I tried a new approach for me, the flipped classroom. I started with a topic for which many students struggled with the homework, and recorded that lecture. During class I worked two examples, then showed them the homework. They spent the remainder of the meeting working the homework problems while I moved around the classroom offering help.

All students claimed they liked this format. Moreover, they performed measurably better. I gave a very similar homework assignment in 2012 as in 2011 and 2010, and the average went up from 88% and 86% to 98% in 2012. The lowest score increased from 65% in 2010 and 63% in 2011 to 90% in 2012. Why? I was able to give individualized instruction to students who really needed it. I will expand the use of this wonderful technique over the next few years.

As part of our SACS accreditation, I recently changed the mission statement for the MS in software engineering program to the following:

> *To teach students to become leaders in engineering high quality, large scale, computing solutions to real life problems.*

This obviously reflects my growing view of my teaching responsibilities as being a leader, but also truly reflects what our students gain. They come in with varying technical skills and job experience ranging from none to decades. They graduate ready to lead their companies with modern skills and practice, to become team leaders, to become project managers, or to lead the way with advanced knowledge and innovative techniques. Ny current view as an educator is that I am preparing leaders.

As I've matured, I have also sought ways to broaden my impact. I used to try to reach individual students, then entire classes. As director of the MS in software engineering, my leadership affects all of our students. When Paul Ammann and I published our textbook, we realized we were expanding our reach out of George Mason to students of software testing throughout the world. This fall I joined an ACM curriculum committee that is updating the recommended curriculum in software engineering, and am suggesting many of the innovations that we implemented at George Mason first. Last month I published an opinion piece that argues that software engineering should be taught more, should be more distinct from computer science, and should look more like traditional engineering disciplines. That conversation has just started; I have almost two dozen messages about this article in my inbox, waiting for responses.

I believe that Mason has done many exciting and innovative things in software engineering education. I only claim partial credit for a few. My current and past colleagues, Paul Ammann, Hassan Gomaa, Sam Malek, Joao Sousa, Ye Wu, Bo Sanden, and others have contributed as much or more. In reality, though, our secret is our students. We have always had terrific students that taught us as much as we have taught them. I close this section with one of my favorite Chinese phrases:

Bu qi xia wen

Loosely translated, that means “don't feel shame when you learn from a person you think is below you.” It works for me because I don’t feel anybody is below me, and I am eager to learn from anyone.

Evidential Inserts for Section 5.5.iii

- Putting the Engineering into Software Engineering Education

  Last fall I wrote this opinion piece for IEEE Software, a widely distributed journal that contains material accessible to all software engineers.

  Local version: http://www.cs.gmu.edu/~offutt/rsrch/abstracts/TeachEngrInSWE.html

- Teaching Principles

  Last fall my department asked me to make a presentation about how I was using technologies in the classroom. As part of that effort, I wrote down some of my guiding principles for teaching. Many of these flow directly from the general Leadership Rules in section 5.5.i. The first is a constant from my first lecture as a graduate student. A couple of years ago one of my colleagues said to me after graduation “I saw you congratulating the software engineering graduates. You really care about them, don’t you?” He was right and the fact that he noticed made me feel proud.

Teaching Principles

1. If you don’t care about them, they won’t care what you say
2. Don’t expect them to be students like you were
3. The main correlation with success is frequent detailed feedback--not class size, difficulty, quality of lectures, book, ...
4. Respect them as people, even if they don’t earn respect from their results
5. Fear is not respect
6. Teach to individuals, but grade anonymously
7. Don’t apologize for doing the right thing, but always apologize for mistakes
8. If you don’t know the answer, help them find who does
9. If they think you want them to learn, they will learn more

http://www.psychologytoday.com/blog/curious/201204/what-will-make-your-kid-succeed-in-kindergarten

Jeff Offutt

April 2012

5.6 The Inspirer

That’s my journey and that’s where I am now. What next? I don’t know. I think I would like to be the kind of person that inspires others to do more, try harder, and be better. How? I don’t know that either, but not knowing what happens next or how to make it happen hasn’t stopped me before.

The software engineering created a minor and a concentration in the Applied Computer Science degree, but I won’t be satisfied until we create a complete BS in Software Engineering. I am sure it will be welcomed by students and industry, but am not sure how we could staff it.

A couple of years ago I started reading literature on educating children who are intellectually gifted. Traditional classrooms and traditional teaching doesn’t work for most of these children, and if they are not taught fast enough or deeply enough, they are at high risk for anti-social behavior, drug and alcohol addiction, poor school performance, and even sociopathic tendencies. The K-12 literature on this subject has been growing, but what do we have for teaching intellectually gifted college students? Very little. One of my current projects is to identify such students and accommodate them. I give them different assignments, engage them out of the classroom, and encourage them to consider special topics classes. Society and universities spend enormous resources on children at the low end of the IQ curve and students with disabilities, but very little on children and students at the high end. Don’t they need just as much accommodation?

Over the next few years I want to do more with online, asynchronous education. I will expand my use of the flipped classroom. I also want to repeat the formula I used in SWE 763 in other classes. And I hope to try the idea of crowd-teaching by involving colleagues at other universities. And this can’t be a solo effort. I hope to encourage more of my colleagues in the Volgenau school in using technology to improve education, not just to cheapen it.

Recording lectures has made me decide that I want my lectures to be short and sweet—10 to 15 minutes. Over the next few years I plan to take my hour-long and 2.5 hour-long lectures and break them into smaller sections. As I found in preparing recorded lectures for my testing class, this sometimes requires creative reorganization of the material.

Finally, although I love SWE 205, I want to fundamentally change it. The material now flows through the two books and is augmented by ideas from other books. But a simpler approach is to define 25 to 50 *principles of usability*, and have each lecture present a principle, then define it, and support it with examples. The principles are there, but not organized. This may lead to another textbook …

Section 6A: Summary of Student Ratings Past 5 years

TEACHING EVALUATIONS (PAST 5 YEARS)

Q1: My overall rating of the teaching

Q2: My overall rating of this course

Course	Q1	Q2	Enrollment
Fall 2012			
SWE 432	4.74	4.63	43
SWE 637	4.63	4.51	21
Spring 2012			
SWE 205	4.94	4.78	18
SWE 763	4.69	4.83	14
Fall 2011			
SWE 432	4.78	4.29	44
SWE 637	4.70	4.35	23
Spring 2011			
SWE 205	4.88	4.63	20
SWE 632	4.70	4.55	24
Fall 2010			
SWE 637	4.57	4.46	36
SWE 642	4.71	4.74	37
Spring 2010			
SWE 205	5.00	5.00	11
SWE 632	4.55	4.41	20
Fall 2009			
SWE 637	4.43	4.46	42
SWE 642	4.53	4.47	39
Spring 2009			
SWE 632	4.48	4.35	29
SWE 642	4.59	4.38	36
Fall 2008			
SWE 637	4.60	4.33	30
SWE 763	4.93	4.71	15
Spring 2008			
SWE 437	4.21	4.23	17
SWE 632	4.52	4.48	35
Fall 2007			
SWE 432	4.74	4.55	32
SWE 637	4.76	4.57	22

Section 6C: Closing Statement

I honestly don’t know how to end this. I’ve already said everything, plus it’s midnight Sunday night and I’ve used up all my words. Putting this portfolio has given me a marvelous chance to reflect on 25 years of teaching, 20 at Mason. I understand far more about my teaching and this reflection will make me a better teacher.

Maybe I should talk about some of my future plans. When I was in college in the 1980s we occasionally got solution manuals with books. In the 1990s, some authors would distribute errata lists, then CDs with examples and sometimes slides. Although our publisher suggested a CD with our book, we opted for a website. We host it ourselves so we can update the documents with improved slides, new errata for the book, and corrections to the solution manual. Although this was innovative in 2008, we’re already thinking further ahead. A book today shouldn’t just come with slides, it should come with fully recorded lectures. Teachers at other universities can send students to the book website for the lectures, and come to the classroom for clarifications and individualized help working problems. This is the next step in my evolution from seeing my job as being a lecturer to being a leader in software engineering education.

Teaching these students at this university is a privilege, made better by the many excellent colleagues I’ve been fortunate to work with. Of course, any award that recognizes hard work is welcome, especially the hard work of teaching that is not recognized during promotions and regular evaluations. I would like to see quality teaching become more emphasized in the Volgenau School, and this award would help me to become an advocate for better teaching and a champion for excellent teachers. I want to expand on my success as a teacher and pass the lessons I’ve learned on to my colleagues. Winning this award would give me the credibility to lead an effort towards more excellent teaching in my school.

I strongly believe that technology should only be used in education if it improves the learning experience for students. The current debate about technology in education is very active, but much is shallow and uninformed. It confuses the economic issue to the point that some expect to use technology only to reduce the cost of education. Some of my colleagues even think their jobs are threatened. My philosophy is that we should only use technology to improve education, not to cheapen it. And after several years of trying different models, I am confident that I have learned how to do just that.

I’ll close by admitting that in assembling this portfolio, I have already gotten the best reward. The letters I received from my students, past and current, brought me untold happiness. I cried, I laughed, and I fondly reminisced about students past. Their success in their careers and lives is the most important award I could ever receive. I dedicate this portfolio to the thousands of students who have passed through my classroom.

If you’ve read this far, I thank you for your consideration, your time, and your attention.