Syllabus for CS 687 Empirical Software Engineering
Fall 2009

Instructor:

Dr. Jane Hayes (www.cs.uky.edu/~hayes).
Room 233, Hardymon Building
Office hours TR 0915 - 1000 (Robotics (CRMS) Bldg, Room 514D)
or by appointment

Course information:

Course homepage http://selab.netlab.uky.edu/homepage/CS687-emp-sw-eng-fall09.htm
Course: CS 687 Empirical Software Engineering
Section: 002
Meets: TR 11:00 – 12:15
Location: Oliver H Raymond Building-Rm.C226-OHR

Description:

The course will present the following: Detailed study of the scientific process; particularly using the experimental method. Examination of how empirical studies are carried out in software engineering (by industry and by researchers). Review of the distinction between analytical techniques and empirical techniques. Study of when experimentation is required in software engineering, and what kinds of problems can be solved using experimentation. Examination of how to control variables and to eliminate bias in experimentation. Examination of analysis and presentation of empirical data for decision making. Students will learn how the scientific process should be applied, how and when to apply it in the software engineering area, and how to evaluate empirical evidence. The principles will be reinforced by examination of published experimental studies, and through designing and carrying out small experiments. On completion of the course, students will be in a position to design and carry out experiments in ways appropriate for a given problem, and will acquire skills in analyzing and presenting experimental data.

Course Outcomes:

Outcome 1 - The student shall know the scientific process
Outcome 2 - The student shall understand and be able to perform experimental design
Outcome 3 - The student shall understand the principles of experimental research and be able to carry out small experiments
Outcome 4 - The student shall be able to critically evaluate the empirical research carried out by others

Course Materials:

Required Text:
Clases Wohlin, Per Runeson, Martin Host, Magnus C. Ohlsson, Bjorn Regnell, Anders Wesslen
Experimentation in Software Engineering: An Introduction
November 1999
Kluwer Academic Pub
ISBN: 0792386825
You must obtain a copy of Wohlin et al.

Other readings, as assigned: See list below.

Course web page:
Course materials will be available on the course web page. The course web page and e-mail will be important methods of distributing information for the course.

Grading:

Your grade in CS 687 will be determined according to these weights:

M.S. students:

Attendance and participation:	10%
Paper summaries:	20%
Team research project:	40%
Presentation:	30%

Ph.D. students:

Attendance and participation:	10%
Paper summaries:	20%
Individual research project:	40%
Presentation:	15%
Lecture Presentation:	15%

Where:

A= 92 - 100%
B = 83 - 91%
C= 74 - 82%
D= 65 - 73%
F = 64 and below

Papers:

The first nine papers are about experimentation, and the rest are descriptions of experiments. It is important that you read the papers BEFORE the lectures, as the discussion will be very interactive. Turn in simple summaries and evaluations of **four** of the first nine **by Tuesday, 9/15/09**. For **one half** of the remaining papers (10), turn in a short (about one page) summary of the paper **by Tuesday 12/1/09**. The summaries should: (1) describe the problem in general terms, (2) paraphrase the experimental hypothesis, (3) summarize and critique the design, (4) discuss the conduct of the experiment, (5) explain whether the hypothesis was proved or disproved, and (6) critique the presentation of the paper. *Paper evaluations will be graded according to the following scale: 0: not submitted, 1: marginal, 2: what was expected, 3: outstanding. You are expected to have read all articles. Proper language usage is required.*

Whining Lowers Grades [1]:

You are always welcome and encouraged to discuss exams and assignments with your professor; it is an excellent way to learn from your mistakes. If the grading does not make sense to you, please ask. You may not yet have understood your mistake -- or there may be an error in the grading. However, whining, demanding a re-grade instead of requesting one, or saying that you deserve more points are good ways to convince a professor to re-grade your entire assignment or exam, perhaps with more careful attention to your mistakes.

Attendance:

Students are expected to attend and participate in all scheduled classes. Arrival after attendance has been taken at the start of class will be considered an absence. The following are acceptable reasons for excused absences: 1) serious illness; 2) illness or death of family member; 3) University-related trips (S.R. 5.2.4.2.C); 4) major religious holidays; 5) other circumstances that the instructor finds to be "reasonable cause for nonattendance." It is the student's responsibility to

Field Code Changed

Field Code Changed

provide sufficient documentation regarding the nature of the absence, and the instructor retains the right to ask for such proof.

Late Policy:

Assignments must be submitted **in person** at or before **class time** on the day the assignment is due. Assignments turned in after class starts are **late**. Credit will be deducted for late assignments. Assignments will not be accepted after solutions have been distributed.

Academic Honor Code:

Individual work (homework, exams) must be your own. No sharing of computer code or other work will be allowed. Group projects allow the sharing of ideas and computer code within the group. No sharing of work **between** groups will be acceptable. The University of Kentucky's guidelines regarding academic dishonesty will be strictly enforced. "All incidents of cheating and plagiarism are taken very seriously at this University. The minimum penalty for a first infraction is a zero on the assignment. [3]" **See attached policy on plagiarism, also here.**

Accepting Responsibility for Failure [2]:

Failure is an unpleasant word, with bleak connotations. Yet it is a word that applies to every one of us at different stages of our lives. No one is exempt. Our icons, gurus, religious leaders, politicians, rock stars and educators all fail. It is simply a reality of being human. It is also a label that we fight desperately to avoid. And it is this fight to avoid failure that drives us forward towards our life accomplishments. So--why can't we take responsibility for our own failure when it does occur?

We need to accept responsibility for a very important reason--namely, maturity. We cannot reach a full level of maturity until we accept ownership of our own mistakes. As an educator, I am confronted with this problem on a daily basis. When a student is late for class, it is because a parent failed to wake them up. A failed test becomes the responsibility of the teacher, the system, society, an after school job, but never the fault of the test taker. An incomplete assignment is inevitably due to the needy demands of a friend, or an electrical failure. I feel particularly blessed because the power circuits leading to my home must be exceptionally fine, as I have yet to experience the myriad of blackouts that have plagued my students.

Nevertheless, the daily onslaught of excuses has left me questioning the value of our education system. What, after all, is the point of "higher learning" if we fail to master the basic task of owning up to our own mistakes?

As we proceed through our education system and indeed life, our excuses for failure become more grandiose and perhaps more grotesque because the crude reality is that we have failed to mature in any significant sense of the word. To continually shift responsibility away from ourselves is worse than being a coward. Even a coward will admit that their failure is a result of their own lack of courage.

Accepting failure takes strength of character, honesty and humility. It provides a building block for future achievements. When we deny culpability, we rob ourselves of the chance to learn from our mistakes. We condemn ourselves to a lifetime pattern of avoidance and deception. Like Marley's ghost, dragging his chains of missed humanitarian opportunities behind him, we crawl forward pulling our chains of pathetic excuses behind us--never fully maturing, never fully reaching our true potential. This stale baggage is far more character eroding than any of our individual failures could ever be.

Computer Facilities:

You will be assigned an account for this course in the Multilab, a PC laboratory administered by the Computer Science department and located in Room 203 of the Engineering Annex, as well as

the CSLab. For information regarding these laboratories, see links under "facilities" from the Computer Science homepage (www.cs.uky.edu). You may use alternative computer systems for developing and testing your work, provided that your submitted work will compile and run under the proper software environment as directed in class.

Group Projects:

The group projects for the course will require you to work together with other students in the class. You will be evaluated on your contribution to the group project and presentations of the project results. The instructor retains the right to make group assignments. Group members are not guaranteed to receive the same grade; evaluation of the group will be individualized to determine individual understanding, commitment, and mastery of the project goals. As part of the project, written reports will be required. Proper language usage is required.

Schedule:

Week	**Date**	**Readings**	**Topics**	**Project, Homework, Exam**
1	Thu 8/27/09	Paper 1	Introduction, Overview of Scientific Method Lecture 1	
2	Tues 9/1/09	Paper 1,2	Lecture 1, Experimentation in Software Engineering, Lecture 2	
2	Thu 9/3/09	Papers 2 – 5, Wohlin Chapter 1, 2	Experimentation in Software Engineering, Lecture 2 – 4, ~~Ethics~~	Hand out project assignment
3	Tues 9/8/09	Papers 6 - 9, Wohlin Chapter 3, 4, 5, 6, 7, 10	Experimentation in Software Engineering, Lecture 5 – 8, **Guest from Ag Center (Dr. Joe Chappell)**	~~Topic selection~~
3	Thu 9/10/09	Papers 6 - 9, Wohlin Chapter 3, 4, 5, 6, 7, 10	Experimentation in Software Engineering, Lecture 5 - 8	
4	Tues 9/15/09	Papers 10, 11, 12, Wohlin Chapter 8, 9	Metrics and Complexity, **Guest from the Writing Center**	**Four summaries due**, Topic selection
4	Thu 9/17/09	Papers 10, 11, 12, Wohlin Chapter 8, 9	Metrics and Complexity, Ethics	
5	Tues 9/22/09	Wohlin Chapter 11, 12	Project Day	~~Experiment Design Reviews~~
5	Thu 9/24/09	Wohlin Chapter 11, 12	Project Day	~~Experiment Design Reviews~~
6	Tues 9/29/09	Papers 13, 14, 16	Testing, lecture Assert-Assess	Experiment Design Reviews
6	Thu 10/1/09	Papers 13, 14, 16	Testing, lecture Assert-Assess	Experiment Design Reviews
7	Tues 10/6/09	Papers 23a, 21, 22	Maintenance, lecture Writing	
7	Thu 10/8/09	Papers 23a, 21, 22	Maintenance, lecture Writing	
8	Tues 10/13/09	Papers 32, 33, 34, 35	Traceability	Hand out sample paper
8	Thurs 10/15/09	Papers 32, 33, 34, 35	Traceability	~~Artifact Review~~

Formatted Table

9	Tues 10/20/09	Papers 23, 24	Requirements & Design	
9	Thurs 10/22/09	Papers 23, 24	Requirements & Design	Artifact Review
10	Tues 10/27/09	Papers 25, 26	Design	~~Draft paper due~~
10	Thurs 10/29/09	Papers 25, 26	Design	
11	Tues 11/3/09	Papers 27, 28	Design, Lecture Presentations	Draft paper due ~~Reviews due~~
11	Thurs 11/5/09	Papers 27, 28	Design, Lecture Presentations	
12	Tues 11/10/09	Papers 29, 30	HCI, Management and Inspections	Reviews due , Hand out sample presentation
12	Thu 11/12/09	Papers 29, 30	~~HCI, Management and Inspections~~ – No class – work on project	
13	Tues 11/17/09	None	HCI, Management and Inspections~~Project Presentations~~	~~Final research papers due~~
13	Thurs 11/19/09	None	~~Project Presentations~~Catch up	
14	Tues 11/24/09	None	Project Presentations	Final research papers due
14	Thurs 11/26/09	NO CLASS Have fun, be safe!		
15	Tues 12/1/09	None	Project Presentations	All reading paper summaries due
15	Thurs 12/3/09	None	Project Presentations	
16	Tues 12/8/09	None	Project Presentations	
16	Thurs 12/10/09	None	Project Presentations	
Final	Tues 12/15/09 1030 - 1300	None	Project Presentations – if time slot needed	

The syllabus is subject to change, and you are responsible for keeping informed of any alterations.

Readings:

Empirical Methods Overview

1. **National Research Council**, *Academic Careers for Experimental Computer Scientists and Engineers, Ch. 1*, National Acadamy Press, pages 9-33, 1994. TOC PS
2. **Fenton**, Norman, Shari Lawrence Pfleeger and Robert L. Glass, "Science and Substance: A Challenge to Software Engineers", *IEEE Software*, V. 11, N. 4, pages 86-95, July 1994. Paper
3. **Tichy**, Walter F., "Hints for Reviewing Empirical Work in Software Engineering", *Empirical Software Engineering*, 5(4):309-312, December 2000. EMSE Home

Field Code Changed

4. **Amschler Andrews,** Anneliese and Arundeep S. Pradhan, "Ethical Issues in Empirical Software Engineering: The Limits of Policy", *Empirical Software Engineering*, 6(2):105-110, June 2001. EMSE Home
5. **Zendler**, Andreas, "A Preliminary Software Engineering Theory as Investigated by Published Experiments", *Empirical Software Engineering*, 6(2):161-180, June 2001. EMSE Home
6. **Harrison**, Warren "Editorial: Open Source and Empirical Software Engineering", *Empirical Software Engineering*, 6(3):193-194, September 2001. EMSE Home
7. **Shull**, Forrest, Manoel G. Mendoncça, Victor Basili, et al. "Knowledge-Sharing Issues in Experimental Software Engineering", *Empirical Software Engineering*, (9)1-2:111-137, March 2004. EMSE Home
8. **Karahasanovic',** Amela, Bente Anda, Erik Arisholm, Siw Elisabeth Hove, Magne Jørgensen, Dag I K Sjøberg and Ray Welland, "Collecting Feedback During Software Engineering Experiments", *Empirical Software Engineering*, 10(2):113-147, April 2005. EMSE Home
9. **Offutt**, Jeff, Yuan Yang and Jane Hayes, "SEEWeb: Making Experimental Artifacts Available", *Workshop on Empirical Research in Software Testing*, Boston, MA, July 2004. PDF

Metrics and Complexity

10. • **L. Briand** and J. Wust, "Empirical Studies of Quality Models in Object-Oriented Systems", *Advances in Computers*, vol. 56, 2002, Academic Press. Briand's homepage
11. • **Fenton**, Norman and Niclas Ohlsson, "Quantitative Analysis of Faults and Failures in a Complex Software System", *IEEE Transactions on Software Engineering*, (26)8:797-814, August 2000. PDF
12. • **Wohlin**, Claes, and Anneliese Amschler Andrews "Prioritizing and Assessing Software Project Success Factors and Project Characteristics using Subjective Data", *Empirical Software Engineering*, (8)3:285-308, September 2003. EMSE Home

Testing

13. • **Juristo**, Natalia, Ana M. Moreno, Sira Vegas "Reviewing 25 Years of Testing Technique Experiments", *Empirical Software Engineering*, (9)1-2:7-44, March 2004. EMSE Home
14. • **Ma**, Yu-Seung, Jeff Offutt and Yong Rae Kwon, "MuJava: An Automated Class Mutation System", *Journal of Software Testing, Verification and Reliability*, 15(2):97-133, June 2005. PDF
15. • **Roger T. Alexander** and Jeff Offutt, "Empirical Evaluation of Coupling-based Testing Techniques for Object-oriented Programs", submitted. PDF
16. • **Lionel C. Briand**, Massimiliano Di Penta and Yvan Labiche, "Assessing and Improving State-Based Class Testing: A Series of Experiments", *IEEE Transactions on Software Engineering*, 30(11), November 2004. PDF
17. • **Grindal**, Mats, Jeff Offutt and Jonas Mellin, "State-of-Practice: An Investigation of Testing Maturity", submitted. Preliminary version
18. • ~~**Stuart C. Reid,** "An Empirical Analysis of Equivalence Partitioning, Boundary Value Analysis and Random Testing", Proceedings of the 4th International Software Metrics Symposium (METRICS '97), 1997. PDF~~

Maintenance

19. • **Kajko-Mattsson**, Mira, "A Survey of Documentation Practice within Corrective Maintenance", *Empirical Software Engineering*, 10(1):31-55, January 2005. EMSE Home
20. • ~~**Liguo Yu**, Stephen R. Schach, Kai Chen and Jeff Offutt , "Categorization of Common Coupling and its Application to the Maintainability of the Linux Kernel", *IEEE Transactions on Software Engineering*, 30(10):694-706, October 2004. PDF local~~
21. • **Kai Chen**, Stephen R. Schach, Liguo Yu, Jeff Offutt and Gillian Z. Heller, "Open-Source Change Logs", *Kluwer's Empirical Software Engineering*, 9(3):197-210, September 2004. online EMSE Home

22. • **Stephen R. Schach**, Bo Jin, Liguo Yu, Gillian Z. Heller and Jeff Offutt, "Determining the Distribution of Maintenance Categories: Survey versus Measurement", *Kluwer's Empirical Software Engineering*, 8(4):351-365, December 2003. online EMSE Home

23a. Hassan, A. E. 2009. Predicting faults using the complexity of code changes. In Proceedings of the 2009 IEEE 31st international Conference on Software Engineering - Volume 00 (May 16 - 24, 2009). International Conference on Software Engineering. IEEE Computer Society, Washington, DC, 78-88 PDF

Requirements

23. • **Damian**, Daniela, James Chisan, Lakshminarayanan Vaidyanathasamy and Yogendra Pal, "Requirements Engineering and Downstream Software Development: Findings from a Case Study", *Empirical Software Engineering*, (10)3:255-28, July 2005. EMSE Home

Design

24. • **Iris Reinhartz-Berger** and Dov Dori, "OPM vs. UML--Experimenting with Comprehension and Construction of Web Application Models", *Empirical Software Engineering*, 10(1), January 2005. EMSE Home
25. • **Marek Vokáccaron**, Walter Tichy, Dag I. K. SjØberg, Erik Arisholm and Magne Aldrin, "A Controlled Experiment Comparing the Maintainability of Programs Designed with and without Design Patterns-A Replication in a Real Programming Environment", *Empirical Software Engineering*, 9(3):149-195, September 2004. EMSE Home
26. • **Anda**, Bente and Dag I. K. Sjøberg, "Investigating the Role of Use Cases in the Construction of Class Diagrams", *Empirical Software Engineering*, (10)3:285-309, July 2005. EMSE Home
27. • **Svahnberg**, Mikael and Claes Wohlin "An Investigation of a Method for Identifying a Software Architecture Candidate with Respect to Quality Attributes", *Empirical Software Engineering*, (10)2:149-181, April 2005. EMSE Home
28. • **Knight**, John C. and Nancy G. Leveson, "An Experimental Evaluation of the Assumption of Independence in Multiversion Programming", *IEEE Transactions on Software Engineering*, (SE-12)1:96-109, January 1986. NEC Research Index (CiteSeer)

HCI

29. • **Miara**, Richard J., Joyce A. Musselman, Juan A. Navarro, and Ben Shneiderman, "Program Indentation and Comprehensibility", *Communications of the ACM*, (26)11:861-867, November 1983. ACM

Management and Inspections

30. • **McDonald**, James, "The Impact of Project Planning Team Experience on Software Project Cost Estimates", *Empirical Software Engineering*, (10)2:219-234, April 2005. EMSE Home
31. • ~~**Thelin**, Thomas, Per Runeson, Claes Wohlin, et al. "Evaluation of Usage-Based Reading - Conclusions after Three Experiments", *Empirical Software Engineering*, (9)1-2:77-110, March 2004. EMSE Home~~

Traceability

32. O.C.Z. Gotel and A.C.W. Finkelstein. An analysis of the requirements traceability problem. In 1st International Conference on Requirements Engineering, pages 94--101, 1994. PDF
33. Antoniol, G., Canfora, G., Casazza, G., De Lucia, A., and Merlo, E. Recovering Traceability Links between Code and Documentation. IEEE Transactions on Software Engineering, Volume 28, No. 10, October 2002, 970-983. PDF
34. Jane Huffman Hayes, Alex Dekhtyar: A Framework for Comparing Requirements Tracing Experiments. International Journal of Software Engineering and Knowledge Engineering 15(5): 751-782 (2005) PDF

Field Code Changed

Field Code Changed

Field Code Changed

35. Jane Cleland-Huang, Raffaella Settimi, Oussama Ben Khadra, Eugenia Berezhanskaya, Selvia Christina: Goal-centric traceability for managing non-functional requirements. ICSE 2005: 362-371 PDF

[1] Dr. Judy Goldsmith
[2] http://www.scs.ryerson.ca/~dwoit/failure.html.
[3] www.uky.edu/Ombud/acadoffenses/letterOfWarningExample.doc