# CS846 Advanced Topics in Software Engineering (0.50) SEM
# Empirical Studies of People Doing Software Engineering
# Fall 2025

Instructor: Daniel M. Berry (dberry@uwaterloo.ca)
Course Web Site: http://se.uwaterloo.ca/~dberry/ATRE/

**Difference Between This Course and Others with Similar Titles**

You may have noticed that other profs, e.g., Mike Godfrey and Mei Nagappan, possibly in other terms, teach other sections of "CS846 Advanced Topics in Software Engineering" with subtitles including the word "Empirical". Each of those courses and this course covers a different flavor of empirical software engineering. Mike's empirical studies tend to be focused on answering specific questions by using the artifacts stored in the data base that is maintained for a software development project. Mei's empirical studies tend to be focused on what can be discovered by mining software repositories. My empirical studies tend to be focused on answering specific questions by conducting experiments with software developers that apply the studied processes in controlled ways. There will be some overlap, as, e.g., a controlled experiment's conclusions might be triangulated with data from artifacts or mined from repositories.

**Course Structure**

I will give the about half of the lectures, covering some basic principles of empirical studies in software engineering, and then going into empirical software engineering work I have done, including experiments, experience reports, grounded analyses of interview data, and critiques of empirical studies.

The rest of the lectures will be given by you, the students, each on a topic of your own choosing. You could do some empirical study of your own, most likely at the scale of a pilot study, you could report on an experience of your own; we could even allow a few of you to conduct a pilot controlled experiment in class. (I hope that you will agree to be a subject in such an experiment. Any such experiments and at least watching them being done in class is considered part of this course.) You could do a thorough study of published empirical studies attempting to answer a specific question of your own choosing.

When I say "of your own choosing", I do suggest that you get feedback in advance of starting, for several reasons: You don't want to discover only when you hand in your report that it was not suitable all along. You don't want to discover that your topic was okay, but you went down a rabbit's hole into something irrelevant. You don't want to discover, near the deadline, that you have no way to finish in time, because you chose too big of a topic, that might be suitable as a thesis topic, but *way* too big for a class project.

**Course Requirements**

Each student will either

1. do some project, e.g., a pilot experiment with volunteering subjects (It is not clear how this would work while we are online!),

2. thoroughly explore the research on some topic involving the way people do SE,

3. conduct some pilot empirical study, e.g., with an anonymous survey or with interviews, of the way people do SE, e.g., at a company for which you have worked or in a more general setting, or

4. explore empirical studies of the way people do work in your own area particularly if this work leads to software's being developed,

and then write it up *and* present it to the class in a 30-minute (the standard conference presentation time) presentation. The amount of time may be adjusted due to the population of the class and the time available

Any of these could end up being the basis for a master's essay or a master's or doctor's thesis.

To avoid potential problems with a too-small or a too-big project, study, or topic, I strongly suggest running it by me before going too deeply in it. Also if more than one want to work on the same project, study, or topic, we can ensure that there will either be cooperation or no overlap.

To encourage people to volunteer to present early, let it be known that the earliest presentations will be evaluated with greater leniency than the latest presentations.

**Possible Topics**

Here, you will find a growing list of possible topics that I think of. These are not intended to be the only possible topics. They are just the ones I think of. I *encourage* you to think of your own. You might end up making a great new discovery worthy of publication!

1. Inspection, including whether and how well it works

2. Agile Methods, including whether and how well they work

3. Formal Methods, including whether and how well they work

4. [Your favorite] Method, including whether and how well it works

5. How to conduct externally and internally valid studies of methods that are intended to be applied to large scale software

**Reading Assignments**

I will be putting papers or URLs pointing to them at the course Web site. Please check periodically for new readings. Please keep up with the reading, as it gives background that I will assume in my talks, and it will help you choose your topic and to do it better.

When putting a *copy* of what I want you to read at the publicly readable course Web site will violate copyright restrictions, I will e-mail you a copy, put a copy at <vault.cs>, or put at the course Web site a URL that works through the UW library site to get you to the original paper. If you are not on campus or logged in via a VPN that makes it appear that you are on campus, you will have to use your watIam credentials to log in to the library site, and once logged in, these URLs will work through the proxy there.

**Course Themes**

The two overriding themes of the class are:

1. How does one conduct a valid test of a hypothesis about how best to develop software?

2. How does one detect that an experiment does not test what it is claimed to test or that an experiment's results do not show what they are claimed to show?