CAS CS411: Software Engineering
2022.FALL.SYLLABUS.1.0

Lecturer
Perry Donham / perryd@bu.edu / 202-567-7362
Mailslot: 111 Cummington / MCS138
Blog: http://sites.bu.edu/perryd Twitter: @perrydBUCS
Office: PSY-228C (64 Cummington Mall)

About the course
While the term *software engineering* covers a broad range of topics, it in general applies to the overall process of creating software solutions. It isn't just about writing code; in fact the implementation of software is usually the smaller part of a project. In CS411 we will take a close look at the various phases of software projects: Definition; Design; Development; Delivery; Management; and Maintenance. In each phase we'll explore current methodologies and see how the pieces fit together into successful projects. Overarching topics include software quality and how to achieve it by applying specific techniques in each project phase, and best practices in architecting software solutions.

After completing CS411 you should be able to understand and apply the basic concepts of software engineering to plan and execute software projects in any phase. You'll also be exposed to a variety of tools currently used in software projects in industry.

Text
Our text this semester is Sommerville's:

Software Engineering, 10th edition, Pearson
ISBN 978-0-13-394303-0 (hardcover)
ASIN B015TKCR92 (Kindle)

It's available at the bookstore (and Amazon, of course), for rental on Kindle, and also is available on CourseSmart in the 9th edition, which is pretty close to the 10th edition. Only off by one, as they say. I don't consider the book to be required.

Delivery Tools
We use Piazza as a repository for the slide sets for each class, copies of homework assignments, sample code, and announcements. You should be enrolled already, so that when you log on to the site you'll see the course listed.

Piazza is also our tool of choice for discussions, including group / project team discussions. We'll use github / gitlab for course and project code and Gradescope, too, for submitting work, because we didn't already have enough tools.

Programming and Prerequisites
In order to give you a chance to practice the process you have learned we'll be working in small teams to design and implement a software project, specifically a web application. At this point in your life you should have some experience in programming (in any language, though my understanding is that you all are Java and Python ninjas…remember 111/112?), but you'll be

using technologies and tools that you might or might not be familiar with, including client-side web page processing with JSP/ASP/PHP, Angular, React, etc.; MySQL, NoSQL and Mongo databases, back-end processing in Python or Ruby or Node.js… basically the MEAN or XAMP stack. My philosophy of teaching is that it's important to learn theory, but even more important to put the theory into practice, so about half of our time will be spent exploring tools and techniques. The good news: No Picobot!

Don't panic if you aren't an expert in everything that we talk about…that's the whole point of you being in class. Good software engineers are what IBM calls 'T-shaped' people…broad knowledge of lots of different things, but with the ability to dive deep when needed. It isn't possible to build robust systems with just a narrow understanding of a few aspects of computing. Anyway, we will work together (and in teams) to figure out things like versioning strategies, proper n-tier architectures, how to normalize database tables, and so on. If you haven't used git before, now you will.

A word about the teams. Typically a small portion of each section comprises non-CS major students, and we see a fair number of folks from Questrom. I do my best to make sure that each project team has a variety of skills, and so if you happen to be a Questrom biz major with no programming skills, you might choose to do some of the project management or UI design aspects of the course, or you might want to learn something about coding or databases. Ditto for CS majors…I don't want you to do all of the coding, so you might find other ways to contribute. Getting experience working in teams is a big part of this course, and each team will need to figure out who is doing what in an equitable way on their own (with guidance if needed).

Tools

Software engineers spend a lot of time learning and using tools and toolchains. I'll be using resources in class that you might find useful, too, and they're free to boot. You might need to poke for the free academic versions of these.

StarUML: An architecture/design tool for drawing UML diagrams. http://staruml.io. Last time I checked there was a non-expiring full demo. I use this occasionally in class to make a point, and there's at least one assignment asking for UML diagrams that you might find StarUML useful for.

Jetbrains (https://www.jetbrains.com) produces high-quality IDEs for various platforms, including WebStorm (Node.js), pyCharm (Python), intelliJ IDEA (Java), PHPStorm (PHP) and RubyMine (Ruby). Visual Studio Code from Microsoft is a fairly good choice, too, if you are in the Windows ecosystem. I know many of you like editors such as Atom or Sublime, but as your project becomes more complex you'll find that these tools fall short of what you need. They are good editors, but not good development environments.

We'll also be using git on github or gitlab; you are welcome to use whatever client (including their web interface) you prefer. I tend to use the command line for git, which to me is the most efficient way to work with repos. We'll go over versioning systems and git specifically in class.

You are of course welcome to use other tools that you might be comfortable with, and we'll from time to time highlight a tool in class.

Grades
There are several grading inputs in the course. We'll have a written exam in the first and second half, and you'll also be working on a project in small teams that will play a significant role in your final grade. There will be several work products (specifications, test plans, and so on) that will contribute to your project grade. Most weeks there will be a short quiz covering the previous lecture's material. Case studies will be assigned during the semester, and you'll be expected to write a short summary of each in addition to participating in in-class discussions.

The allocation for your final grade looks like this:

Midterm exam	25%
Quizzes	30%
Cases	15%
Team project	20% (all team members receive the same grade)
Team 360º review	10%

The grading scale is numerical:

96-100	A	80-84	B	65-69	C
90-95	A-	75-79	B-	60-64	C-
85-89	B+	70-74	C+		

If your course grade happens to be close to a boundary, such as an 89.5, I'll bump you up unless your overall course work for some reason doesn't justify it. For quizzes, I'll drop your lowest-scoring quiz when doing final grades. There are no make-ups for quizzes.

Team Project and 360º Reviews
Teams are formed in the first few weeks to work on a project together. The project parameters are contained in a series of documents / assignments posted on our course site. All team members will receive the same grade for their project. Broadly speaking, what I'm looking for in project work is that your team follows a design and development process in a consistent way, using tools and techniques that we discuss in class.

In addition to the team grade, each team member will receive a 360º Review grade. For the 360º Review, each team member will evaluate the participation and contribution of the other members of the team. We do this so that if only one or two team members do all of the work, the non-working members don't get a free ride in terms of the project grade.

Lab / discussion sections
In addition to lecture time, each of you has an assigned discussion section. We'll use that time in a few different ways. Early in the course, it will be a place to read cases, do quizzes, and any other individual assignments that come along. Project teams will be formed from within the discussion sections, and once work on projects begins the discussion section will be used for teams to work together on their project, with feedback and guidance from your TF or me.

Contacting me and office hours
The best way to contact me is by email. You may also text me at perryd@bu.edu (iMessage).
My office is in the Psychology building at 64 Cummington Mall, room PSY-228C. Office hours

will be posted on Piazza. No appointment needed, just drop by if you have a question or want to hang out a bit. If you need to drop something off, my mail slot is in the CS office in MCS-138.

Academic Conduct Code

The University and the College take cheating very seriously. Cheating and plagiarism will not be tolerated in any course. Cases will be referred to the Dean's office and may result in loss of credit for an exam or assignment or other disciplinary action.

By nature, programming is a collaborative effort, and I fully expect that you will use resources such as Google, fellow students, and our own discussion forum on Blackboard to learn the material and do your assignments. We'll discuss code and approaches in class, and I'll occasionally post sample code on Blackboard that you are welcome to use as a starting point. However, I definitely don't want you to simply copy entire programs that you find on the web and turn them in as your own work. If you do use more than a line or two of someone else's code (including mine), just make a note in a comment in your program to point to where you got it.

Dr. Sullivan's guidelines for collaboration are a good explanation of our expectations. You can read them at http://cs-people.bu.edu/dgs/courses/cs111/collaboration.html.