Informal Computer Engineering Overview for Peer Advisors and Incoming Students

Robert Dick

This document briefly summarizes topics of interest to peer advisors and students who are considering joining the Computer Engineering major. It has a few embedded URLs.

1 What is Computer Engineering?

Computer Engineering students learn the theory and practice of designing and analyzing complex digital hardware/software systems. They learn how to design computers from bottom to top. The program covers transistors, logic gates, processors, operating systems, and applications. The focus is on creating digital systems to make life better.

2 How Does Computer Engineering Relate to Electrical Engineering?

Computer Engineers focus on digital systems, i.e., systems in which discrete values are used to represent state (on and off, integer numbers). Although the transistors they work with are analog components, they use simplified models that treat them as either on or off. Simplifying the models of individual components allows Computer Engineers to design working systems with millions of components. Computer Engineers must be good with digital circuits, logic, microprocessors, and software. This allows them to design complete hardware-software computer systems.

Electrical Engineers generally use more complex component models than Computer Engineers. For example, they may represent the current through a transistor as a non-linear, gradually varying function of gate voltage while Computer Engineers represent it with a step function. Working with more complex, often analog, component models generally means that Electrical Engineers are limited to designing systems with fewer components. Electrical Engineers are not required to be good with each part of computer system design. This allows them to focus on a particular topic such as signal processing, control theory, or analog circuits.

3 How Does Computer Engineering Relate to Computer Science?

Many concepts learned by Computer Engineers and Computer Scientists overlap. For example, both learn about algorithms, computational complexity, and software design. However, they learn about these things for different reasons. There are many reasons a Computer Scientist might learn these things. A Computer Engineer learns them to use in designing and building complex hardware-software computer systems. In addition, Computer Engineers learn about circuits, logic, and microprocessor architecture. If you want to design, analyze, and build complex hardware-software systems, pick Computer Engineering. If you want to focus on the theory or practice of analyzing, using, or designing parts of computer systems, e.g., designing artificial intelligence algorithms without necessarily understanding the details of hardware used to run them, pick Computer Science.

4 What do Computer Engineers do After Graduating?

Many go to the many companies working on designing embedded systems, i.e., special-purpose computers such as those in smartphones, cars, and robots. Many go into software engineering. Some go into microprocessor design. A few go into management consulting. It is a fairly flexible major because recruiters looking for software engineers generally interview Computer Scientists and Computer Engineers and recruiters looking for digital hardware engineers generally interview Computer Engineers and Electrical Engineers. Pay and job opportunities for all three majors fluctuate a bit from year to year, but are generally similar. In short, it is probably best to pick the major (among these three) you are most interested in because the differences in pay and opportunities usually aren't huge.

5 What Classes Should I Take Next Semester?

Based on my time as a Computer Engineering advisor and teacher, I believe that this algorithm is effective for choosing courses.

1. Have you taken EECS 270? If not take it. If so, go to the next step. There is some correlation between liking EECS 270 and liking Computer Engineering. EECS 270 is also a prerequisite for many other Computer Engineering courses. Take it early. There may be exceptions, e.g., waiting for a professor you prefer or taking another course early because it is required for an intership you would like.

2. Open the Computer Engineering Program Guide to page 6. Read the course descriptions for all of the "ULCE" courses. Tentatively select three, at least one of which (an MDE) must be in a dark box. Then work backwards in the dependency graph to find the courses you should be taking this semester. I recommend this approach for two reasons. First, some of the lower-level courses are more abstract, building a foundation for later work. It is generally easier to figure out the upper-level courses you will be most interested in. Second, many of the upper-level courses require 35–40 hours per week. Knocking out the prerequisites early gives you the flexibility to start taking heavy upper-level courses earlier so you don't need to take too many during any single semester.

3. Verify that you aren't taking too heavy a load with the help of the EECS Workload Surveys.

6 When Should Students Interested in Computer Engineering Talk With Computer Engineering Advisors?

As soon as they have significant interest in the major. That way, we can explain the major and make sure they are taking appropriate classes to be on track for later declaration. It is easy to schedule an appointment here.

7 The Computer Engineering Program Guide

The Computer Engineering Program Guide is a terse description of the major. Students who are seriously considering the major should read it. Pages 1–4 and 10 cover essential administrative information (what you need to do to graduate). Pages 5–9 provide advice to make it easy for students to determine their main interests and career goals and select appropriate courses. I don't personally see much value in Page 7, because many good schedules differ from this example. Page 5 may be valuable to students who are just entering the major, but you will most likely gather the same knowledge by reading all the Upper-Level CE Elective course descriptions.

Please send suggestions of improvements to this overview to Robert Dick <dickrp@umich.edu>.