Winter 2019 Midterm Instructor Report of EECS 398-001: Special Topics for Nicole Hamilton

Project Title: **Winter 2019 Midterm Evaluation**

Course Audience: **61**
Responses Received: **15**
Response Ratio: **24.6%**

Report Comments

This report is a summary that tabulates all quantitative ratings on a single page. Comments appear at the end of the report. Ratings are from the Winter 2019 midterm teaching evaluations of EECS 398-001: Special Topics.

Prepared by: **Office of the Registrar**
Creation Date: **Sunday, March 3, 2019**

Responses to questions related to the course:

	SA	A	N	D	SD	N/A	Median
I had a strong desire to take this course.	9	4	2	0	0	0	4.7
This course advanced my understanding of the subject matter.	7	5	2	1	0	0	4.4
My interest in the subject has increased because of this course.	10	1	3	1	0	0	4.8
I knew what was expected of me in this course.	3	2	6	3	1	0	3.1
Work requirements and grading system were clear from the beginning. (Q232)	1	5	5	1	3	0	3.2
Overall, this is an excellent course.	3	5	4	1	2	0	3.6
I am learning a great deal in this course.	3	8	2	2	0	0	3.9
The amount of work required so far appears to be appropriate for the credit being received.	4	6	4	1	0	0	3.9

Responses to questions related to the instructor:

	SA	A	N	D	SD	N/A	Median
Overall, Nicole Hamilton is an excellent teacher.	4	2	4	2	1	1	3.4
Nicole Hamilton gives clear explanations.	4	4	2	3	1	0	3.8
Nicole Hamilton acknowledges all questions insofar as possible.	10	3	1	0	0	0	4.8
Nicole Hamilton uses class time well.	3	6	3	3	0	0	3.8
Nicole Hamilton seems well-prepared for each class.	3	5	3	3	1	0	3.6
Nicole Hamilton uses techniques to foster class participation.	6	5	2	1	1	0	4.2
Nicole Hamilton treats students with respect.	8	4	2	0	1	0	4.6
Nicole Hamilton is teaching in a manner that serves my needs as a student.	6	1	4	2	2	0	3.4
Nicole Hamilton is willing to meet and help students outside class.	9	5	0	0	0	1	4.7
Nicole Hamilton is enthusiastic.	10	3	1	0	0	0	4.8
Nicole Hamilton keeps students informed of their progress.	6	6	2	1	0	0	4.3
Nicole Hamilton sets high standards for students.	10	4	0	1	0	0	4.8

Written Comments

What are the major strengths of this class? What is helping you to learn? (Q979)

Comments
I would say the major strengths of the class are that the project is interesting and lecture has provided a lot of the programming and design information necessary to accomplish it. I think the homework/ lab assignments have been fun and not overly complicated, which is good because we also have a large project to work on.
The parts of lectures that focus on teaching concepts (e.g. index) are useful for the project and interesting.
I think working through a large design project with other members is the major strength of this course.
Labs
I'm not learning from the class... it's more my own research instead.
The labs and homework have been incredibly helpful so far in reinforcing the concepts taught during lecture. Shout–out to Carolyn for creating such an important and helpful aspect of this class.
There is a lot of room for creativity with all of the assignments and the big project. Professor Hamilton and the IAs are very knowledgeable and willing to help students.
The biggest strength of this class is the concept of the class itself. We are building a (relatively) complex software system. This fact alone means that we are learning many of the core, practical skills that really good engineers master. We deal directly with the OS (networking, threads, processes, memory, disk I/0), have to deal with data on a (relatively) massive scale, heavily analyze performance and trade–offs, write lots of code, and, perhaps most importantly, work in a team of realistic size and composure, where each member has variable skill levels and experiences. I think that a class that offers all of this – and does at least a decent job on every part – is an incredible opportunity for undergrads who are just starting their careers in software. All of these aspects of the course are what makes it a truly special offering at UofM – I am extremely grateful I have the opportunity to take it.
It almost feels like 485 in that it teaches different aspects from different parts of computer science.
This class is a great way to combine knowledge learned in other classes or if not having taken them you get a good intro to them.
The multithreading lab and other homework projects have been very helpful in learning concepts better. The emphasis on planning the project has also been helpful.
The professor, Nicole Hamilton is a superb lecturer. I very much appreciate her industry experience as an engineer on software projects. That one of her advantages, as well as this class's advantages over other classes.

How can Nicole Hamilton improve this class? If possible, give specific examples. (Q980)

Comments
I like that she is always willing to answer questions and that she does so really thoroughly; however, I feel like between people asking a lot of questions and going over both Windows and Linux code for every topic, we sometimes don't get to everything we want to cover in a given class period and end up going over more important topics too quickly. Just paying more attention to pacing would help.
I think EECS 482 should be required for this class. Maybe about half the class already took EECS 482, so almost the entire first half of the semester wasn't that useful for those people. I also think I would learn more if more time in class was spent teaching concepts and less on showing specific examples. Much of the examples seemed to take more time than was necessary to teach a particular concept, and even then I would sometimes get lost in the specifics of the examples. Maybe the more in–depth examples could be converted to homework. I also think that the class should focus on teaching for just 1 operating system. Doing both Windows and Linux examples is just too much and doesn't seem useful for the majority of people. It would make sense to choose Linux just because that is what most people studying CS at umich are used to, but I would even rather just do Windows than dividing my attention between the 2.
Not sure what I should be getting out of each lecture
1. Be prepared for class. *THIS ONE IS TRULY OUTRAGEOUS* Why would I ever be prepared for class if you aren't? This is the most unprofessional thing a teacher can do. I have 0% belief that you'll have your midterm done before the 28th. 2. Don't ask "who doesn't know?" because that shames students who don't. Ask, and have a student explain to foster participation. 3. Make slides that are actually helpful to learning. Blocks of text are a waste of time, better formatting, include graphics that can help. (You can also tell they were cheaply put together simply in black and white, just... no effort put in. If a student did this for a presentation, they'd lose a ton of points. Why shouldn't a teacher do the same?)
We spend some time during lecture reviewing the material covered during the last lecture. While this review is good as a reminder of what we were talking about, sometimes we spend too much time on it.

Comments
The course still seems very disorganized and the grading scale for assignments seems to be arbitrary and sometimes defined after the assignments have been submitted.
Before I start, I want to say I think this is a great class and I think that you're a great instructor for it. However, I do think there are aspects of the course that could be improved to make it even better. So, here are my critiques, in order of how important I think changing them is. 1. Far too much class time is spent going over source code in class. I mostly mean that up to half the class is each day is going through code that illustrates the use of various OS calls on both Windows and Linux. I think that this can be resolved very simply – have the entire class based on a Linux system. There are a few reasons for this. First and foremost, many (if not a vast majority) of students taking the course have zero experience programming in a Windows environment. All of the intro courses and the majority of upper level EECS courses are in Linux environments. Given the number of other things students are expected to learn in this class, learning Windows system calls is simply not important relative to the others. Another reason for this is that if the class was based purely in a Linux environment, it would give students the opportunity to learn one OS, and learn it (somewhat) deeply. I know that you come from a Windows background, but to me its pretty obvious you have learned more about the Linux environment that 99% of the students in the class. 2. The use of C++ in the class doesn't make sense. The name of the course is System Design in C++. Yet, the code we are required to write is heavily discouraged from using the standard library of the language. I am aware that the STL has its flaws, but use of C++11, 14, and 17 features (which are, in a very large way, compartmentalized into the STL) is a major part of what C++ developers working on production projects are expected to use. Over the past years, the performance and consistency of the STL has been massively improved, and I am worried that your demonization of it will convince many in the class that the current C++ STL is garbage (which it is not) and dissuade them from learning it. However, I actually agree with the goal you are trying to accomplish by barring its use – the goal of getting students to write a system, by and large, on their own. So my recommendation is to use C for the course. This recommendation is similar to my point about using Linux for the environment. Rather than having people write heavily restricted and archaic C++ code, have them write it in C. They will learn how to use the tool (C) deeply (and in a manner that more closely represents what production C code looks like), rather than essentially writing C code with some classes thrown in and run through a C++ compiler. 3. The final issue worth touching on is the pre–requisites for the course. I understand that one of your main goals is to give students a real experience that teaches them real skills that they can really talk about early on in their education. But as a second semester senior in the course, I know for a fact that I would not be prepared if I had taken this course much earlier in my career. At a minimum, you should require 370. You talk about bits, hardware, processes, etc. all of the time in lecture, but a lot of this material has next to 0 coverage in the courses up to 281. Having 370 under their belt would give students at least some grasp of the low–level concepts that you stress in class and that are essential to understand for the engine. There are obviously highly motivated students who already know about most of these things before being in 370 or their upper levels, but if your goal is to provide the best experience as possible for students, then having them be properly prepared is important. If I was in charge I would require 482 as well, because that would enable the class to have much more focus on the design of a great software system (which in my opinion is what is really important) rather than explaining the basics of how an OS works, but I don't think its as important as having 370 and I know you probably won't ever want to require 482.
Explanation of some topics is very unclear Instructor uses technical terms and assumes everyone understands. Please do not add requirements to homework without letting us know, you did not specify that lines of code was a factor for grading and later took off points for that in HW1. Forces us to implement everything from scratch. I think it would be more appropriate to leave it to us what we should re–implement for optimization
I think Hamilton needs to be more straightforward with teaching concepts. For instance, sometimes she'll use jargon, then ask if anyone knows what the word means, and then give a definition for it, but doesn't really tell us how to apply that jargon (ex: the lecture where she talked about a CAT program). Something that would have been more useful more would be like 'this is the definition of a CAT program. Here's what it does, then tell us it's called a CAT program, and tell us what it can be useful for in our search engine project.
It would greatly help me if grading rubrics could be more clear. If we are not allowed to use vectors or if lines of code will be counted, that should be announced prior to submission. The rubric should not change after the assignment is due. Secondly, It should be clear how you will be counting lines of code. If we are only required to have taken 281 to be prepared for this class, you should address when you are going to stray from how the rest of the EECS department run things. Also, I would appreciate if lecture slides could explain topics other than just providing example code. All of the example code on canvas is uncommented!! this is not helpful when you are trying to understand a concept and the additional resources can't confirm your interpretation of how the code works.
I have no idea how to study for the midterm or how we will be evaluated, so more clear expectations and idea of "testable material" would be helpful.

Comments
None, it's superb.

Please enter any additional comments you have for Nicole Hamilton. (Q981)

Comments
I think she has done a really great job, and I've enjoyed the class a lot so far. The lectures have been well–prepared and relevant, and her lecture style is conversational and enthusiastic and engaging. She is always willing to chat or answer questions. The course has also been really organized for being in only its second semester. She's crushing it.
Lots of excellent experience and wisdom to share
How did you get hired here to teach? You're literally the most incompetent teacher I've ever met. To be labeled as a teacher is a shame to all teachers that actually try to help students learn. There is no preparation, there is no creativity in your teaching, and there is no professionalism. And I still despise you for your comment in the CS engineering page about slowing down and EVERYONE making 100k after graduation. That is simply pathetic. And I'm sure you don't know how to handle feedback effectively too, so this is just great. Just because you're knowledgeable doesn't make you a good teacher.
This class is incredible so far. I'm thoroughly enjoying the class even though it's still relatively new and unpolished. The lectures are very interesting and informative, the labs are very engaging, and the project as a whole is a wonderful challenge. This class has done wonders for my resume because I finally feel like I have something interesting and unique that I can confidently talk about with recruiters. Thank you for offering this class and making it such a great success!
I know it's only the second semester it's been offered, but I think this course has the potential to become as popular as EECS 482.
This has been an awesome experience so far and I truly think that this class, with some changes and more time under its belt, will be the most valuable course for students at UofM. Additionally, I am a massive fan of your focus on the Unix basis for much of what we know today. I think that more time in the class should be dedicated to exploring this type of stuff, as I don't think any of my other classes have even touched on the individuals and concepts that laid the foundation for the incredible things we are doing with software today. THANK YOU FOR TEACHING THIS CLASS AT UM!!!
Explanation of some topics is very unclear Instructor uses technical terms and assumes everyone understands. Please do not add requirements to homework without letting us know, you did not specify that lines of code was a factor for grading and later took off points for that in HW1. Forces us to implement everything from scratch. I think it would be more appropriate to leave it to us what we should re–implement for optimization so that we can focus more on the overall design
Sometimes I feel that we are not using class time effectively if there are 112 slides posted and we only make it to 54. Does this mean we are now behind? Should I not be worried about it? It concerns me that we won't get through the necessary information and teams that have members that already know this information will have an unfair advantage to others. Since you've demonstrated that you grade on a steep scale when you say competitively, I would want the initial playing field to be even. Barely getting through the information before that milestone is due does not provide that.
I think the competitively graded photo should be worth less points or graded non–competitively, since every team submitted a creative and high–quality group photo, yet several teams still received low scores.
I admit I disagree a bit with the professor's policy on being discouraged from using the STL. But I know she must have good reasons for doing so, because I trust and respect her. Professor Hamilton is superb at getting the class involved in class. She absolutely cares about if someone is a little bit confused about a concept or a topic, and will try hard to get a meaning across. She admittedly seems to care more about her students than the professors I have had so far, but that might be because most of my computer science classes literally have 200 people enrolled. This is one of the few classes computer science classes where the instructor actually knows my name. I have no regrets having professor Hamilton as a professor.