University of Utah
School of Computing

Course Staff and Schedule

Instructor	Joe Zachary
	Office: 3190a MEB
	Phone: 581-7079
	Email: zachary@cs.utah.edu
	Office hours: MW 1:15-2:45 p.m.
Lectures	Tuesday/Thursday; 2:00-3:20 p.m.; L101 WEB
Discussions	Wednesday; 8:35 a.m., 9:40 a.m., 10:45 a.m., 11:50 a.m., 12:55 p.m.; 3225 MEB

Learning Goals

Large software systems are the most complicated artifacts that have ever been built. Successfully building large software systems typically requires organizing the efforts of a number of people, knowing and following a variety of technical strategies and processes, and exploiting a collection of computer-based tools. Students who are relatively new to programming can be oblivious to the organizational and technical challenges posed in creating large systems, because approaches that succeed for small software projects often don't scale and can be disastrous for larger projects.

The field of software engineering encompasses the study and application of effective tools and techniques for organizing the efforts of teams of people towards designing, developing, deploying, and maintaining software systems. The steps in creating a software system are

- determining the requirements for the system (by interacting with the system's clients),

- designing the system (by analyzing the requirements and describing the system as a collection of modules and components),

- constructing the system (by making effective use of programming languages, tools, and techniques), and

- testing the system (by developing test suites for all levels of the system).

In addition, a system must be maintained over the course of its lifecycle, which may require revisiting aspects of the system's requirements, design, implementation, and testing.

Software engineers have developed radically different methodologies for accomplishing these four steps. At one end of the spectrum are Waterfall processes, in which the participants attempt to address each step completely before moving along to the next: the requirements are completely determined, then the system is completely designed, then the system is completely implemented, and finally the system is completely tested. Each step in a waterfall process, applied to a large system, might consume months. There is a premium on getting things right the first time, since discovering during testing that changes must be made to the requirements can be an expensive proposition. As a consequence, there is an emphasis on management hierarchies and written reports.

At the other end of the spectrum are Agile processes, in which the participants iterate repeatedly through the four steps, with the goal of gradually growing the system. Each iteration incorporates some new aspect of the system and produces an incomplete but usable system that serves as the basis of the next iteration. Individual iterations require only a matter of weeks. As the system grows, it can become necessary to reconsider prior design decisions and refactor existing code. There is a premium on frequent and rapid interactions among clients and developers. As a consequence, there is an emphasis on face-to-face interactions among all the individuals involved.

CS 3500 is an introduction to software engineering. Although you will learn about all the steps involved in creating a system, we will focus primarily on the software construction step by building on the programming background that you acquired from CS 1410 and CS 2420. CS 3505 will in turn build on this course by treating the rest of the software development process in more depth.

In CS 3500, we will bridge the gap between introductory programming and full-blown software engineering by addressing these four major topics:

- C# programming using Visual Studio

- Software construction techniques using a variety of tools

- Software architecture

- Working individually, in pairs, and in small groups on projects ranging in size from small programs to large systems

C# Programming. A portion of the course will be devoted to studying C#. All students should have had an introductory programming course (such as CS 1410) and a programming-intensive course in algorithms and data structures (such as CS 2420). Most of you will have learned Java in these two courses; others of you may have studied different programming languages such as ActionScript and C++. Regardless of the particular language(s) you have used, you should be proficient in these topics:

- Program development (using a development environment to create, compile, test, and debug programs)

- Implementing classes (understanding member variables and their role in information hiding, constructors and their role in initialization, methods and their role in abstraction)

- Implementing constructors and methods (using method calls, assignments, conditionals, and loops)

- Using and implementing common algorithms (e.g., sorting and searching) and data structures (e.g., stacks, queues, lists, and trees)

- Exploiting class libraries (especially those that provide common algorithms and data structures)

- Understanding techniques for exploiting polymorphism (via inheritance, interfaces and templates)

If you do not have this background, you should not take CS 3500. I will spend time helping you transfer your prior knowledge in Java (or whatever language) to the context of C#, but this presupposes that

you have that prior knowledge! In addition, we will spend time expanding your knowledge of C# in particular and of programming in general.

Software Construction Techniques. As you learn about and practice with C# over the course of the semester, you will be doing lots of programming. As you program, you will confront a number of issues in software construction that are not specific to C#. We will discuss these issues in lecture and will expect you to apply what you learn in your C# programs. These issues include

- Gathering requirements

- Designing programs

- UML (Unified Modeling Language)

- Object-oriented design patterns

- Coding style

- Source code control

- Scripting

- Code inspections

- Testing

- Debugging

- Performance profiling

Software Architecture. Over the years, a variety of approaches have evolved for decomposing a program into pieces and then arranging for those pieces to communicate. These approaches are known as *software architectures.* If you are familiar with some of the standard software architectures, you will become a much more proficient programmer.

I will be illustrating different software architectures via classroom examples as you learn C#, and you will get practice with them via programming assignments. Some of the architectures that I plan to explain are:

- Pipe and filter. A series of programs arranged into a pipeline, where each program accepts input from its predecessor, modifies it, and passes the results along to its successor. This is among the oldest of the common software architectures.

- Object-oriented. A program composed of a collection of classes whose interactions are mediated by a type system that exploits inheritance and parametric polymorphism. You should already have a fair amount of experience with this architecture.

- Client/server. A system partitioned into a centralized server and a collection of remote clients. Users interact with clients, which communicate over the network to obtain services from the server. The key issues here will include using network sockets for remote communication, using threads to ensure highly-available servers, and using locks to coordinate the threads.

- Event-driven. A system that reacts to events generated by an external agent. A good example of an event-driven system is a graphical user interface, which reacts to mouse and keyboard events.

- Model-view-controller (MVC). An interactive system that separately maintains the representation of the data being manipulated (the model), the rendering with which the user interacts (the view), and the rules for dealing with user interactions (the controller). Web servers are typically organized along these lines.

Group Work on Large Systems. We will spend roughly the first two-thirds of the semester studying C# programming, software construction techniques, and software architecture. During this time, you will working individually on small to medium sized projects. In the last third of the course, we will turn our attention to the issues that arise when working in groups on large systems. Five weeks is not enough time to study such an important issue, however, so this topic will spill over into CS 3505.

You will learn about group efforts and large systems by joining with other students to work on a final project with weekly deadlines. You and your partners will be making modifications to an existing program with a large code base.

Course Overview

Prerequisites. The prerequisites for this class are an introductory programming class (such as CS 1410) and a programming-intensive algorithms and data structures class (such as CS 2420). See the previous section for a discussion of specific topics with which you must be familiar.

Text. There is no text that you have to buy. Instead, I will be assigning reading from a variety of sources on the web.

Hardware/Software. Instructional computing in the College of Engineering is managed by the CADE Lab. Even if you do all of your work for this class on your own computer, you will need to have a CADE Lab login name and password to access the computers in the lab sections, to set up a required SVN repository, and to submit your problem sets. If you don't have a CADE account, visit the operators in WEB 224.

As a CS or CE major (or CS 3510/5010 student), you will have access to the SoC Instructional Lab in WEB 130/124. WEB 130 is full of Windows PCs, whereas WEB 124 contains space where you can use your own laptops.

As a student in the College of Engineering, you also have access to the Windows PCs in the Engman Lab in WEB 210.

You will be using Microsoft Visual Studio 2010 to do your C# development. This development environment is installed on the PCs mentioned above, and you can obtain it at no charge to install on your own computer. Visual Studio is available from the MSDN Microsoft Academic Alliance (MSDNAA). Under this program, you can obtain nearly all systems-related Microsoft software. If you wish to get software under MSDNAA, please send an email request to `opers@eng.utah.edu` and include your CADE login name.

The download for Visual Studio is several gigabytes. If you'd rather not download such as a large file, you can obtain a DVD containing Visual Studio from the opers in the Engman Lab (210 WEB).

Lectures. We will meet for lecture on Tuesdays and Thursdays in WEB L101 from 2:00–3:20. I will use a combination of the chalkboard and a laptop during lectures. I will use the laptop to present programming examples and perform demonstrations; I make limited use of Powerpoint. I will post the laptop-based examples on the class web page, but you will need to take notes if you want to keep track of what I write on the board.

Lab Sections. There will be a laboratory section each Wednesday, *except for August 25*. You should attend one of the six sections, which meet 8:35–9:25 a.m., 9:40–10:30 a.m., 10:45–11:35 a.m., 11:50–12:40 p.m., 12:55–1:45 p.m., and 2:00–2:50 p.m. in MEB 3225. In the sections you will get hands-on practice with tools and techniques under the supervision of the teaching assistants. The lab activities will be

posted on the class web page. If you do not complete an activity during the lab, you should complete it shortly afterward. Lab activities are not graded.

Consulting. All of the course staff (instructor and teaching assistants) will be available outside of formal classes to answer your questions and help with problems. We will post the consulting schedule on the class web page as soon as it is finalized.

Reading. There is a link on the class web page to a schedule that will give the reading assignment and lecture topic for the next class. After each lecture, I will update this schedule to reflect what was actually covered in lecture that day and to show the reading assignment for the following lecture. The schedule will also have links to problem sets, solutions, and projected lecture material. By the end of the semester, the schedule will contain a record of everything we covered.

Problem sets. Most Wednesdays I will post a problem set on the class web page. Each problem set will consist of a collection of written and/or programming problems and will be due six days later on the following Tuesday via electronic handin.

Exams. There will be a midterm and a final. The midterm will be on October 26 in place of lecture, and the final will be on Friday, December 17 from 1:00-3:00 p.m. in the lecture room.

Grading. A weighted average will be calculated based on your problem set average (50%), your midterm exam grade (20%), and your final exam grade (30%). Grades will be assigned on a curve consistent with the weighted averages of the students.

Getting Help and Information

The class web page is `http://www.eng.utah.edu/~cs3500`. This page will be updated frequently with calendar updates, projected lecture materials, problem sets, solutions, lab section exercises, and helpful links. Please check it frequently.

The class mailing list is `cs3500@list.eng.utah.edu`. All registered students are automatically subscribed to this list. The list uses your official UMail address, so be sure that you're reading (or forwarding) any mail that is sent to that address. We will use the class mailing list to send urgent messages, such as corrections to problem sets or changes in due dates, to everyone in the class. You will not be able to send mail to this list yourself.

The staff mailing list is `teach-cs3500@list.eng.utah.edu`. Each member of the course staff will receive a copy of each message that is sent to the staff mailing list. We will reply directly to each question, and we will post the answers to frequently asked questions to the class web page.

We encourage you to seek us out whenever you need help, advice, or encouragement. We will always be available during our regular office hours, and you can make appointments for other times. Simple questions can often be answered by phone or e-mail. Our consulting schedule will be posted on the class web page as soon as it is finalized.

Cooperation vs. Cheating

Working with others on problem sets is a good way to learn the material and we encourage it. However, there are limits to the degree of cooperation that we will permit.

On individual problem sets, you must limit your discussions with other students to a high-level discussion of solution strategies. Anything that you hand in, whether it is a written problem or a computer program, must be written in your own words. *If you base your solution on any other written solution, regardless of the source, you are cheating.*

On group problem sets, of course, you may collaborate with your partners.

When taking an exam, you must work completely independently of everyone else. Any collaboration here, of course, is cheating.

We do not distinguish between cheaters who copy other's work and cheaters who allow their work to be copied.

If you cheat, you will be given an E in the course and referred to the University Student Behavior Committee. If you have any questions about what constitutes cheating, please ask.

Students With Disabilities

Reasonable accommodation will gladly be provided to the known disabilities of students in the class. Please let the instructor know of such situations as soon as possible. If you wish to qualify for exemptions under the Americans With Disabilities Act (ADA), you should also notify the Center for Disabled Students Services, 160 Union Building.

Tentative Class Schedule

Date	Activity	Event		Date	Activity	Event
T 8/24	Lecture 1			T 10/19	Lecture 15	PS 7 due
W 8/25	No Discussion	PS 1 out		W 10/20	Discussion 8	
H 8/26	Lecture 2			H 10/21	Lecture 16	
T 8/30	Lecture 3	PS 1 due		T 10/26	Midterm Exam	
W 9/1	Discussion 2	PS 2 out		W 10/27	Discussion 9	PS 8 out
H 9/2	Lecture 4			H 10/28	Lecture 17	
T 9/7	Lecture 5	PS 2 due		T 11/2	Lecture 18	PS 8 due
W 9/8	Discussion 3	PS 3 out		W 11/3	Discussion 10	PS 9 out
H 9/9	Lecture 6			H 11/4	Lecture 19	
T 9/14	Lecture 7	PS 3 due		T 11/9	Lecture 20	PS 9 due
W 9/15	Discussion 4	PS 4 out		W 11/10	Discussion 11	PS 10 out
H 9/16	Lecture 8			H 11/11	Lecture 21	
T 9/21	Lecture 9	PS 4 due		T 11/16	Lecture 22	PS 10 due
W 9/22	Discussion 5	PS 5 out		W 11/17	Discussion 12	PS 11 out
H 9/23	Lecture 10			H 11/18	Lecture 23	
T 9/28	Lecture 11	PS 5 due		T 11/23	Lecture 24	PS 11 due
W 9/29	Discussion 6	PS 6 out		W 11/24	Discussion 13	PS 12 out
H 9/30	Lecture 12			H 11/25	Thanksgiving	
T 10/5	Lecture 13	PS 6 due		T 11/30	Lecture 25	
W 10/6	Discussion 7	PS 7 out		W 12/1	Discussion 14	
H 10/7	Lecture 14			H 12/2	Lecture 26	
T 10/12	Fall Break			T 12/7	Lecture 27	PS 12 due
W 10/13	Fall Break			W 12/8	Discussion 15	
H 10/14	Fall Break			H 12/9	Lecture 28	
				F 12/17	Final Exam	(1:00-3:00 p.m.)