# Course Overview

Below is the catalog description for CS 3505, Software Practice II. Note that this is a rather vague description, and slightly inaccurate:

An in-depth study of traditional software development (using UML) from inception through implementation. The entire class is team-based, and will include a project that uses an agile process.

More clearly, the purpose of this class is to provide you with the information and skills to work on software projects in a team setting all the way from inception to final product. CS 3500 focused on individual software skills and mostly touched on implementation, debugging, testing, and maintenance. This class will dovetail with 3500 and focus on the parts of software development up to implementation including requirements gathering, requirements analysis, software architecture, and design. An important part of
this class will be to learn about the Unified Modeling Language (UML). We will cover parts of the two main process models: traditional, incremental, structured software development and agile.

During the semester, students will continue to practice their programming skills using C# in Visual Studio 2008. A few projects will be individual assignments, a few will be pair programming assignments, and the remainder will be team projects. Our focus will be on simulations, games, and programs that stress user interactions.

The prerequisites for this course are the successful completion of CS 3500, CS 2420, CS 1410, CS 1010, and MATH 1210. You should have a solid foundation in computer programming, computer science, mathematics, and application development to be enrolled in this course. We will begin assuming that all students know how to create applications in Visual Studio 2008. Students who did poorly in any of the prerequisites should consider re-taking the prerequisite before taking this class.

Students will be expected to solve their own language, application, and technical problems throughout the semester. The teaching staff will be happy to help, but you may not rely on our timely responses or expertise to complete your work.

## Topics

The following is the current topic plan:

- Introduction
- C#
- Gaming
- XML
- Software Process and UML Nuts and Bolts
- Overview of Requirements and Architecture
- Use Cases
- Software Architecture
- OOD and UML Class Design
- Dynamic Modeling
- Agile
- Continuous Integration
- Teams and Team Player Type Analysis
- How to Run Meetings / Thoughts on Management
- Wrap up

The actual topics may vary as I adjust this course to my teaching style.
Pragmatics

Lectures / Discussion Sections

We will meet for lecture on Tuesdays and Thursdays for 80 minutes in WEB 104. In lecture I will exclusively use a laptop for slides, examples, and programming. All notes, examples, and programs will be posted on the class website.

We will hold a limited number of discussion sections this semester on Wednesdays in MEB 3225. Most weeks will not have a discussion section, though, and this time may be reserved for team meetings with a TA.

Attendance will be periodically recorded to help in giving participation marks. Students are expected to attend most classes and discussion sections.

Book

We will use *UML Distilled, Third Edition* during the middle weeks of the semester. Other relevant books and on-line references will be posted on the class website.

Students should find a C# and XNA reference that they like. I recommend online books - I will post links to two good books that are electronically available through the Marriott library.

Projects

There will be fifteen projects due throughout the semester. The release and due dates for each project will be posted throughout the semester on the class website. The topics are as follows:

1. Warm up simulation using C# and forms *Individual*
2. Develop a simple C# XNA application that lets you control a sprite in the game. *Individual*
3. Turn the game into multi-player game. *Individual*
4. Build a more complicated game. *Pairs*
5. Create a networked version of one of the two games. *Pairs*
6. Form your team. *Team*
7. Develop a use case model of the project. *Team...*
8. Analyze the submitted use case models, choose your top three.
9. Develop a design for the project.
10. Build, test, and demo your project solution.
11. Vote on the best project solution to be used for the base of the second project.
12. Phase 1 of second project (methodology shaping, blitz planning, burn chart)
13. Build the walking skeleton for the second project.
14. Build the second increment for the second project.
15. Build, test, and demo your second project solution.

Projects will involve a significant amount of programming and design. As this is a three-hour class, I expect students will spend an *average of 6-8 hours a week* working on the assignments.

Please make sure to review the course schedule regularly. It will be updated with projects and solutions as the semester progresses. I reserve the right to alter the projects and schedule as needed to ensure a successful class.
As this semester stresses teamwork and communication, please note that your solutions to assignments may be made visible for other students to see (with your identification included). Please do not hand in anything objectionable or sensitive.

Unless otherwise stated in the homework assignment, all homework solutions will be due by whatever time is listed in the assignment. The time that we use for verifying a homework is the creation time of the source file in the handin directories. Be careful not to overwrite a file and wipe out its creation time.

Assignments will not be accepted late with the following exceptions:

- **Individual assignments** - Each student will receive two late days for the entire course. Each late day allows you to extend the assignment deadline for one day (24 hours). To use a late day you simply submit the assignment late. We will assume that any assignments submitted late count against your late days.
- **Team assignments** - Each team will receive two late days for the entire course. The same rules apply as above for individuals.
- **Any late assignment that is not accompanied by a valid late-day usage will not be accepted.** There are no exceptions or excuses; the late days are intended to cover unforeseen circumstances. Use them wisely.

No late days can be used for the last project due during finals week.

**Teams**

Special consideration is always required when working in teams. You should select teammates whose understanding of the material is roughly the same as yours. Teams should also be formed based on compatible schedules. Finally, I recommend that all teams make use of the CADE lab machines for the convenience and safety of team members.

During the team phase, if a team member flakes out and doesn’t do the work assigned to them, then we reserve the right to adjust the scores accordingly. This may mean adjusting the score of the person not doing their job lower and possibly, in rare occurrences, adjusting the rest of the team higher (because they had to cover for the person “flaking out”). If this starts to happen at any point during the semester, please inform me immediately.

**Exams**

We plan on scheduling short, in-class quizzes as a means of you demonstrating your understanding of the various class concepts. Depending on how the quizzes are going, we may have a midterm or two.

Exams and quizzes will be posted on the class schedule.

**Grading**

Homework and projects will be worth 70% of your grade, participation will be worth 5%, and quizzes and midterm exams will be worth the remaining 25% of your grade.

Should you discover what you think is an error in grading, you have only TWO WEEKS after the scores are issued to request a regrade. The procedure you should use is to first go and see the TA who did the grading and see if you can get it resolved. If you are still unsatisfied, then you should come and see me.

**Computer resources**

All of the programming that you do in this course will be done using C# in Visual Studio 2008. Visual Studio is available for download by CS students if you have an MSDNAA account. To get an account, see the MSDNAA link on the computer science support page: [http://www.cs.utah.edu/support](http://www.cs.utah.edu/support)
I caution students that installing, updating, and managing Visual Studio takes a considerable amount of time. Please complete assignments in the CADE lab if you have trouble with your own computer and software.

Getting Help and Information

The class website is:

http://www.eng.utah.edu/~cs3505

It will contain a variety of information resources, including course staff consulting hours and e-mail addresses; links to course handouts and problem set solutions; links to examples from lecture, and links pertaining to the textbook, Java, and programming environments.

The course staff email address is:

teach-cs3505@list.eng.utah.edu

This is the correct place to send any questions you may have. Any email you send to this list will go to all of the teaching staff (the instructor and the TAs). Please only send messages to me or your TA individually if privacy is required.

Please read your email regularly. Important announcements will be emailed to the class mailing list. If for some reason you do not receive the class emails, please make sure our emails are not being filtered by your Internet service provider. Also, make sure your official University of Utah email address is correct.

The course staff (instructor and teaching assistants) will hold regular consulting hours each week during which we will be available to help you with questions or problems. The consulting schedule may vary and the current schedule will be available on the class website. We encourage you to seek us out whenever you need help, advice, or encouragement.

Cooperation vs. Cheating

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

When working on individual assignments, you must work only with others whose understanding of the material is approximately equal to yours. In this situation, working together to find a good approach for solving a problem is cooperation; listening while someone dictates a solution is cheating. You must limit collaboration to a high-level discussion of solution strategies, and stop short of actually writing down a group answer. Anything that you hand in, whether it is a written problem or a computer program, must be written from your own thoughts and in your own words. If you base your solution on any other written solution, you are cheating.

When working as teams, each team must not share code or designs with other teams. Teams must solve problems independently.

Handing in code or designs from a different semester (or job) is forbidden and is also considered cheating.

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 others’ work and cheaters who allow their work to be copied.

If you cheat in any amount, you will be caught, and 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. Remember, if you base your solution on any other written or group solution, you are cheating.

Each year, approximately five students who cheat get an E in my classes.