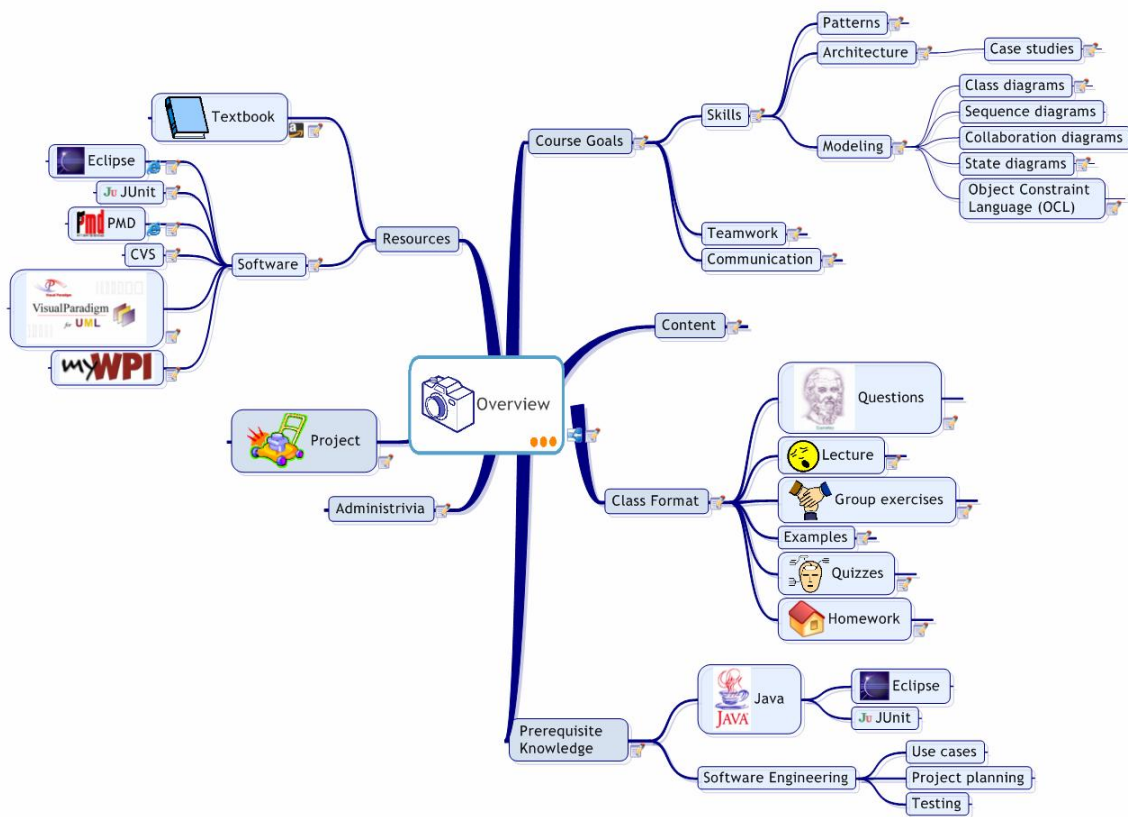


Overview



See document: [CS4233 Object-Oriented Analysis and Design.mmap](#)

Every student should look this section over carefully. There are several reasons for this:

- You need to know what to expect and how much time you should plan in your schedule to successfully complete the course.
- You need to know what you're expected to know coming into the course. Although there are no hard prerequisites, the section on What You Should Know lays out what you should be very comfortable with before taking the course.
- You need to know what you're going to get out of the class (other than a grade) and determine if it fits your overall education plan.
- You need to know how the class will be run, the rules and regulations, and how you'll be graded.
- You need to know who to contact and how to contact them when you need help.

1 Course Goals

What good is taking a course unless you know what you're going to get out of it. At WPI professors are encouraged to identify a small set of *outcomes* that students should realize when they have successfully completed the course. The outcomes and goals for this course are presented in this section.

1.1 Skills

One of the things you're probably most interested in is the set of skills you will acquire, or improve by taking this course. You will certainly improve your overall object-oriented skills. You will have ample opportunity to analyze problems, design solutions to those problems, and implement the solutions using Java.

There are other specific software engineering and O-O skills you will sharpen as well.

1.1.1 Patterns

Patterns have become a major area of study, research, and development in the O-O community today. New patterns are being identified as new techniques and technologies emerge. You will learn details of common patterns, how to write a pattern description, and most importantly how to know when to apply patterns to your code.

1.1.2 Architecture

Architecture is critical to any significant software project. In this class we will discuss many architectural mechanisms and principles and see how they can be applied.

Case studies

We will look at one or two architectural case studies and critique them.

1.1.3 Modeling

While you should know the fundamentals of modeling; especially using UML to represent software, you will learn more details of UML and how to apply the UML to different situations.

Class diagrams

We use more detailed class diagrams than in CS3733. These will include packages.

Sequence diagrams

Collaboration diagrams

State diagrams

State diagrams will be used to describe complex behavior of the state and activity of specific classes and methods.

Object Constraint Language (OCL)

We will use OCL to formally specify behavior of the classes in our models.

1.2 Teamwork

Software is rarely developed by individuals. You will work as part of a team. This course will give you more opportunity to hone your interpersonal skills that will be valuable for your success in the software industry.

1.3 Communication

Being able to communicate well goes hand-in-hand with the ability to work as part of a team. In this class you will have the opportunity to improve your written and oral communication skills.

2 Content

The content of the course consists of several areas. The main content is taken from the text book. However, we will augment this with a couple of extra readings from recent journal articles on OOAD.

3 Class Format

This should be an interactive class. I don't want to spend 50 minutes, four times a week talking to a bunch of sponges. I want the course to be a dialog. There are a few ways that I'm going to try to stimulate the course to help encourage discussion and vary the approach to the material.

3.1 Questions

One of the best ways to really learn something, besides teaching it to others, is to answer questions. The questions may be ones you can readily answer, or they may be ones that you will have to do some research in order to answer them.

I plan on using questions to spark the dialog. This Socratic approach will be a main method of keeping the class moving.

3.2 Lecture

There will be some standard lecture, but I do not plan on using a lot of PowerPoint slides as I have for some of my courses.

3.3 Group exercises

During approximately one class each week we will spend some time on group exercises. The exercises might involve designing a specific class, or set of classes, drawing a UML diagram, or some other exercise that will improve your OOAD skills.

3.4 Examples

We will review a lot of examples of good and bad Object-Oriented systems. During these reviews you will be challenged to identify the appropriate characteristics and defend your choices.

3.5 Quizzes

Once a week, usually on Friday, there will be a short (5-10 min.) quiz on some material we've covered in the last week.

3.6 Homework

Yes, there will be homework. Much of it taken from the exercises in the book. One of the criticisms I got last year was that there were too few homeworks, and the ones that were there were too *big*. So, this year I'll have more smaller ones.

4 Prerequisite Knowledge

There are two main things you need to know in order to be able to manage the technical content of this class:

- You should be proficient in Java programming.
- You should have the skills expected of someone who has successfully passed CS3733.

4.1 Java

4.1.1 Eclipse

4.1.2 JUnit

4.2 Software Engineering

4.2.1 Use cases

4.2.2 Project planning

4.2.3 Testing

5 Administrivia

See the course web page for class policies, staff, office hours, grading, and other information.

6 Project

The project for this class is the analysis, design, and implementation of the "brains" for an automatic lawn mower, called RemoteMower. You will be provided with the framework that simulates the hardware, and you are expected to develop the logic that drives the machine. You will be graded on the architectural qualities of your design, the efficiency and maintainability of your code, and its performance.

Each team is responsible for the complete project. You do not have to interact with other teams for this project.

See the class project web page for more details.

7 Resources

7.1 Textbook

See document: [103-6047726-1411046](#)

This is the primary text for the course. Cay Horstmann is a well-known author and teacher of O-O methods.

7.2 Software

You will use several programs during this course. Each of these is described with information on where to get them. All of the software you need will be available in the computer lab machines as well.

7.2.1 Eclipse

See document: www.eclipse.org

Eclipse is the Java IDE that I expect you to use. It contains everything you need to be an effective software developer for building Java applications, including integration with tools to test, manage changes, and package your work.

The link shown takes you to the Eclipse home page. Currently, Eclipse 3.0 is the recommended choice.

7.2.2 JUnit

JUnit is the unit testing package / framework that you will use to test your work. It is integrated into Eclipse and comes with the standard Eclipse Java development tools that is installed by default.

7.2.3 PMD

See document: pmd.sourceforge.net

PMD is a static code style checker. It is configurable and useful for finding different types of stylistic defects in your code. PMD is available as an Eclipse plug-in. It is installed in the laboratory machines with Eclipse. If you are using your own systems, you will need to install PMD from its SourceForge download site. See the [PMD home page](#).

7.2.4 CVS

CVS is the recommended, but not required, version control software. You will be able to set up a CVS project on the computer science machine. If you do not have a CS account, notify the staff.

7.2.5

Visual Paradigm for UML is the UML modeling tool we will use for this course. It is available on the lab computers and can be downloaded to your own system. The license key for the academic license for the Standard Edition 3.1 will be posted on myWPI.

7.2.6

You're expected to check myWPI regularly for announcements and discussions.