Software Design and Documentation

Syllabus Spring 2018

Mon./Thurs. 4:00pm – 5:50pm CSCI-4440-01 (53104) -02 (51368)
Mon./Thurs. 6:00pm – 7:50pm CSCI-4440-03 (53105)
Walker 5113

John Sturman (sturmj@rpi.edu)
TAs: John Angel (angekj3@rpi.edu) and Yingyi Wu (wuy21@rpi.edu)
207 Lally 518-276-8412

Send all mail to sddteaminstructor@gmail.com

Website: http://sites.google.com/site/rpisdd

Introduction

This is a course about communication. The first step is for you to learn how to communicate a real-world problem and propose a feasible solution. The second step is to transform the real-world representation into diagrams and documentation that will allow your development team to clearly understand and coordinate on designing the solution. The third step is to implement the design and to verify through testing that you indeed solved the real-world problem using the proposed solution.

We will start by giving you the vocabulary and tools that are currently standard for object-oriented design. It will be up to each of you to use the tools and vocabulary to improve your own personal development practices as well as create a more effective team environment.

Learning Outcomes

At the end of this course you will

- Be able to envision, design, develop, and deliver a full software product with a team.
- Be able to describe and use design patterns in the design and development of your software.
- Be able to design and iteratively develop software using a use case-driven approach.
- Gain valuable experience working cooperatively in a group with a common goal using a proven system.
- Be able to coherently describe the differences between various software development

processes in use today.
- Improve your confidence and ability to deliver a sophisticated technical presentation.
- Be able to read and draw simple UML diagrams.

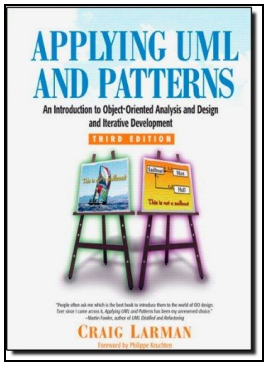
Topics

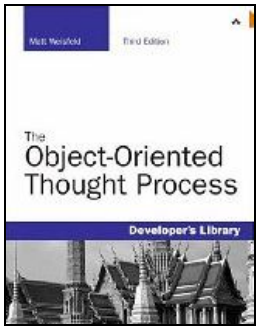
In this course we will discuss the following topics

- Software Development Processes (Waterfall, Unified Process, Scrum, XP, and other Agile approaches to software development)
- Object Oriented Design Theory
- Use Cases
- Design Patterns
- UML
- Software Testing
- Development Practices
- Project Management

Texts

Note that all texts for this course are optional

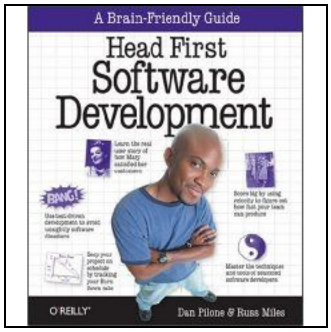
- 

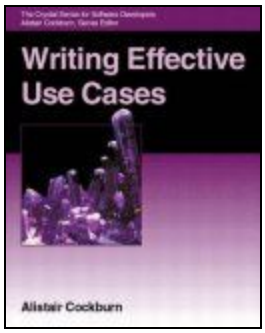
Applying UML and Patterns: An Introduction to Object-Oriented Analysis and Design and Iterative Development (3rd Edition) by Craig Larman ISBN: 0131489062
- 

The Object-Oriented Thought Process (3rd Edition) (Developer's Library) by Matt Weisfeld ISBN: 0-672-33016-4

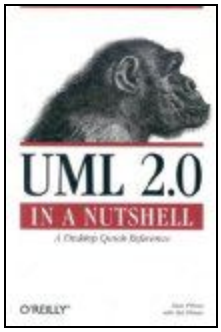
- 

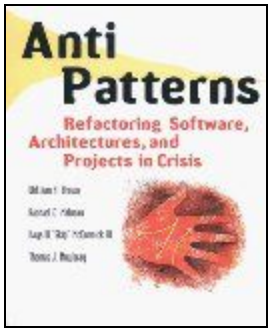
[Design Patterns: Elements of Reusable Object Oriented Software by Gamma, Helm, Johnson, and Vlissides. ISBN: 0-201-63361-2]
(Strongly Recommended)
- 

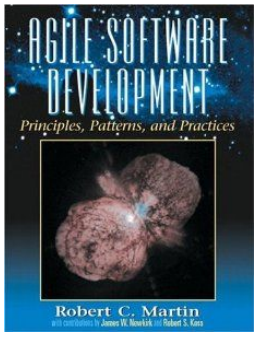
[Head First Design Patterns by Elisabeth Freeman, Eric Freeman, Bert Bates, Kathy Sierra, ISBN: 0-596-00712-4]
- 

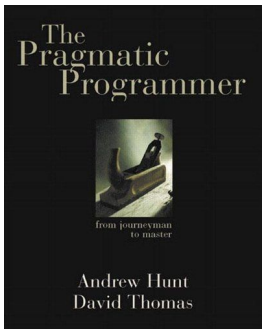
[Head First Software Development (Brain-Friendly Guides) [ILLUSTRATED] (Paperback) by Dan Pilone (Author), Russ Miles (Author) ISBN: 0-596-52735-7]
- 

[Writing Effective Use Cases by Alistair Cockburn. ISBN: 0-201-70225-8]

- 

UML 2.0 in a Nutshell by Dan Pilone, Neil Pitman, ISBN: 0-596-00795-7
(Strongly Recommended)
- 

AntiPatterns: Anti-Patterns, Refactoring Software, Architectures, and Projects in Crisis. By Brown, Malveau, McCormick, and Mowbray. ISBN: 0-471-19713-0
- 

http://www.amazon.com/exec/obidos/ASIN/0470423676/catskillmou0c-20s
Agile Software Development: Principles, Patterns, and Practices. By Robert C. Martin. ISBN: 0-13-597444-5
- 

The Pragmatic Programmer: From Journeyman to Master. By Andrew Hunt and David Thomas. ISBN: 0-20-161622-X

*By purchasing the books using the above links a small portion of your regular purchase price benefits the Catskill Mountain Beekeepers Club.

Many of these books and others are available online to the RPI community at http://proquest.safaribooksonline.com/

Grading

There are two major parts to an individual's final grade: personal and group. The personal portion of your grade will account for 45% of your final grade with the group portion accounting for 55%.

There is no final exam in this course; the group project grade will be the final grade you receive. Enjoy finals week and study hard for your other courses, as SD&D will be over by then.

This course will not be graded on a curve. The grade you earn is the grade you will receive.

There is no extra credit per-se in this course, as you can see the components of your grade come up to exactly 100%. Very low project grades can be improved through a team "fess up and fix" session or by providing an additional presentation. See below for details. While we make every effort to ensure fair and equitable grading, you may dispute any grade directly with the instructor. The grade will be modified only if appropriate.

Portion	Item	Percentage
Group	Project Inception	8.00%
	Project Elaboration	8.00%
	Project Stakeholder Product Review 1	8.00%
	Project Stakeholder Product Review 2	8.00%
	Project Transition	8.00%
	Final Presentation	8.00%
	Best Practices	7.00%
Individual	Oral Pattern Presentation	10.00%
	Take Home Exam	20.00%
	Peer Evaluation	15.00%
Total		100.00%

Group (Project) Portion

With each phase deliverable each team is to include:

- **Project Status Report**

  This is a summary of the state of the project. It should include an updated list of issues encountered thus far. For each issue, include a brief summary of the issue, the person responsible for resolving it, and what measures they are to take. Also include any outstanding risks and the impact it can have on the project. Please summarize progress so

far in the entire project up to the point of the deliverable, not just provide your daily reports.

- **Contribution Summary**

  A listing of how each team member contributed to the specific deliverable. This is to account for each member taking an active role in each aspect of each deliverable.

Project Inception

- **Vision Statement**

  The vision statement, also known as a project charter, should include:

  ★ An executive summary of the project,
  ★ A thorough description of the proposed project which includes details that will interest potential stakeholders,
  ★ A business case for the project describing what benefit this project will provide for the stakeholders, and/or the user community. Include an analysis of the competition, market space, or similar commercial off the shelf software.
  ★ A description of the specific project stakeholders (don't forget Team Instructor).
  ★ A description of the major features of the completed project, and
  ★ A description of the major risks of the project.

- **User Scenarios**

  Create two scenarios that describe a user in your target audience using your software to satisfy one of their goals. Include a description of each persona, details of their goals, illustrative details of the user interface with which the user interacts, and the sequence of steps they go through with the requisite system responses to accomplish their goal.

- **Project Schedule**

  A detailed schedule of the phases and iterations of your project. You must include iterations and dates for the development of each use case or feature, as well as estimates of the time required to complete each item on the schedule. The schedule must show the **specific dates each phase and iteration begins and ends**. Also include preliminary details and timing of the tasks included in completing your project.

Project Elaboration

- **Domain Model Diagram**

  Create a diagram that identifies conceptual classes, associations, and attributes within your project domain. The diagram must use proper UML notation for a domain model diagram.

- **User Stories**

  A collection of the user stories developed for your project. Each user story should include the story itself, notes from discussions and other thoughts, and how to test the particular story. The more information you include the more useful the artifact will be.

- **Supplemental Specification**

  Include a complete set of requirements for your project. Use the FURPS+ model to find any and all requirements for your project. The functional requirement list must include requirements drawn out of the use cases. All requirements must be **numbered** and have

an **assigned priority** using a ranking system such as *MoSCoW*.

- **Deployment Diagram**

  Provide a diagram that is representative of the hardware and software platform(s) used to deploy your solution. Please include descriptions of all required hardware and software and justification for your choices. Proper UML notation is required.

- **Use Cases** (80% complete)

  80% of all known use cases must be written out in detail using the fully dressed format, as discussed in class. Update the complete list of features.

- **Work Breakdown Structure (WBS)**

  A detailed list of the tasks associated with your project based upon the vision, scope, and user scenarios. The list of tasks should be as granular as possible (work package level) so that the team can clearly see everything that needs to be accomplished in order to complete the project.

- **Updated Project Schedule**

  An updated schedule that includes all tasks found in the WBS with specific duration and start and completion dates for each task. Make sure to include all milestones as well as phase and iteration dates and a list of the features that will be delivered for the two Construction sprints.

Project Construction – Project Stakeholder Product Review #1 (Sprint 1)

This is a visual review of the product being developed. After this deliverable date, you will have a few more weeks to complete the development of your project. The following deliverables will be tied to this portion of your grade:

- **Iterative Release**

  An iterative release of your product that includes the promised portions of your feature set for Sprint 1. 100% of your promised functionality for this Sprint must be completed, matching the requisite use cases you developed in Elaboration. This review will be done in class with Team Instructor and any other stakeholders you have identified in your Inception and Elaboration documents.

- **Design Approach**

  A paragraph that describes the design of the code implemented by the team (**must be object-oriented). The paragraph should include the details of design patterns employed as well as any other techniques used in the development of the code.

- **Sequence Diagrams**

  Sequence diagrams showing how a specific Use Case is realized within the beta version of the software. The diagrams must use correct UML notation. Provide at least two diagrams to receive credit for this portion of the deliverable.

- **CRC Cards**

  A set of CRC cards for all the major classes in your project. Include the responsibilities and collaborators for each class as well as a rating of the coupling and cohesion for each class based on class notes. The cards can be presented in a single document.

- **Static Class Diagram**

  A comprehensive static class diagram of the project's internal software structure. If necessary you can break this diagram up into parts to make it more readable. The diagram must be valid UML and accurately reflect the source code.

Project Construction - Project Stakeholder Product Review #2 (Sprint 2)

- **Beta Release**

  A feature-complete beta release of your product. 80% of all your “Must Have” requirements (or the equivalent) must be completed for Construction. Your preliminary test results will need to show proof of the extent of your implemented and tested functionality. You must schedule a demo for the instructor in class. You need to submit a full version of your final developed code for review. (Links to repositories are NOT sufficient.)

- **Code Review**

  Each team is required to submit their code to a comprehensive review by another team and to perform a review on another team’s code. The guidelines of this review will be covered in class. Each team should submit summaries of both the performed and received review. Make sure to name the groups being discussed.

- **Testing Documents**

  The testing documents consist of three parts:

  - Test Plan: A document specifying the goals of your testing effort as well as the resources to be used and the testing schedule.
  - Test Case Specification: A detailed set of step-by-step instructions your testers follow to complete each individual test. Every step and expected result should be included for each test.
  - Test Results: A template for your testers to use to record the results of the tests. Use this template to record (and submit) the results of your Alpha test results.

Project Transition

- **Final Release**

  A tested, final release version of your software. This version should be installable using a method that makes it easy for a new user to get started using your software. Teams are expected to demonstrate their final release (including installation) to Team Instructor during the last week of class.

- **Final Test Results**

  The final results of testing your software against the test documents you delivered during Construction. Include a statement indicating the readiness of the software for public consumption. This is a document which can be used by a stakeholder to determine if the final version of your software is ready for release (or not).

Other Final Deliverables

- **Best Practices Evidence**

  Provide evidence of project team Best Practices. See Best Practices below. This portion of

the Transition deliverable does not figure into the Transition grade, however without any evidence, we provide no credit. This is an all-or-none grade.

- **Peer Reviews** (*individually by each team member*)

  Provide reviews for each person on your team based upon the metrics determined by the team at the beginning of the semester. See Peer Reviews below. This portion of the Transition deliverable does not figure into the Transition grade, however, if you do not submit a peer review for your teammates, you will not receive credit for this portion of your grade. Please submit your reviews in any format you like, specifying each individual's score for each metric as well as a total for each person. Do NOT include a review of yourself. Any comments you would like to include will be welcomed and read.

Final Presentation

At the end of the semester each project group will give a presentation to the class describing their project, its benefits, and the software design your team used to develop it.

Best Practices

This portion of your grade is pass/fail; your group will either earn the credit or won't earn the credit. As such, we encourage you to use as many of these development practices as possible to ensure you will earn the grade.

To get this grade you must implement items 1, 2, 3, and 4 as well as three (or more) others listed here. You must provide documented proof that you have done so. It has been our experience that development teams (both in this course and outside in professional settings) have a higher success rate when they make use of best practices, then when they don't. Evidence of these practices need to be submitted with your Transition phase deliverables to receive credit.

Mandatory Best Practices (must use all)

1. **Project Management Web Site**

   We recommend the use of websites such as GitHub (https://github.com/), CodePlex (http://www.codeplex.com/) or SourceForge (http://www.sourceforge.com) to manage your project. These sites generally encompass several of the functions we require as best practices. While they are very useful, they may not include everything your team may need. Feel free to mix and match as needed. If you use one of these sites as a best practice, you must implement the various features of the site to get credit. For example, using a GitHub site without taking advantage of the issue tracking does not count.

2. **Code Repository**

   Using a code repository is worth the time and effort required to set it up and manage it. When working with more than one person, it is a very good idea to use a repository to hold and control any code being developed. The attributes of a good repository include always

being available and requiring some form of authentication for security and user/usage tracking. The system usually maintains a record of all check-in and check-out activity and keeps a running record of changes made to any files. The system should have the ability to track back versions and perform code merging to synchronize versions of files. Repository examples include Subversion (https://subversion.apache.org/) and Git (https://github.com/).

3. **Documented Coding Standards**

   Documenting the coding standards used by your group can improve the quality of the software that you write by making it more consistent between developers. Coding standards should describe conventions used by the team such as where braces are positioned, naming conventions for classes, methods, variables, and documentation conventions. Many programming languages have several different published coding standards for that language. Your team may wish to adopt a published standard for the language of choice, rather than trying to develop your own. Upon the adoption of coding standards, all written code should be consistent with these standards.

4. **Use the basic concepts behind object oriented design**
   Polymorphism, high cohesion, low coupling, and data encapsulation.

Other Best Practices (must use three or more)

5. **Build Tools**

   Build tools such as make, autoconf, libtool, Ant (http://ant.apache.org/), NAnt, JAM, etc. are essential in large projects as they provide a repeatable means of converting the software from source code form to executable code. DOS batch files or UNIX shell scripts will not be accepted as build tools, as these tend to not work very well in professional settings.

6. **Unit Test Tools**

   Unit testing tools such as JUnit (for Java) or NUnit (for .Net) are very helpful to a project to perform automated testing which is quickly repeatable to verify software quality.

7. **Use Mock Objects in unit tests**
   Mock objects are used to simulate the behavior of objects in a lightweight controlled manner. One example of using mocking is to remove the dependency on database connectivity and database state for unit testing. http://code.google.com/p/moq/

8. **Use Dependency Injection in conjunction with unit tests**
   One example of a DI container is Unity Application Block (Unity)
   http://msdn.microsoft.com/en-us/library/ff647202.aspx.

9. **Bug Tracking System**

   A bug tracking system such as Bugzilla can provide a central means for a project team to report on bugs found within the software, the status of the fixes for those bugs, and a general "to-do" list which is shared across all team members. Generally a bug tracking system is only useful for larger projects, but the instructor has found them to still be very useful in small projects. Github Issue tracker

10. **A Linting System**

    An application that runs through your source code to find formatting discrepancies, non-adherence to coding standards and conventions, and pinpoints possible logical errors in your program. Using a Lint program on your source code, helps to ensure that source code is legible, readable, less polluted and easy to maintain. Some available linters out there include JSLint, CSSLint, JSHint, Pylint.

11. **Third Party Component or Tool**

    Include in your project a component or a tool from another open source project or framework. This included code must be used and attributed according to the guidelines laid out in the license for the software.

12. **Design Patterns**

    Use one or more design patterns to enhance the effectiveness of your project. You may use any pattern discussed in class or any other documented pattern that you find. However, for patterns found outside of the class curriculum, instructor approval is required. You must document the design pattern(s) in your source code and provide an explanation of how and why you used the pattern(s).

Fess Up and Fix

Project teams may earn up to 20 additional points on a late submission or a very low grade on a deliverable by making a 15-minute presentation to the class describing:

- How the deliverable was flawed, or how late it was relative to the planned delivery date.
- What events caused the flaw and/or late delivery.
- What corrective actions the team is taking to make the materials correct again, and by when they will have corrections completed.
  This completion date will form a new milestone for the team. (And it must be met to be eligible to get points back!)
- What actions the team will take to prevent the late delivery and/or serious flaw from occurring in future project milestones.

Up to 20 points will be awarded back at the end of the semester if the presentation conveyed an understanding of the root cause of failure, a corrective course of action, corrected materials were submitted, and the corrective course of action plan is implemented well enough such that the remaining project milestones are met.

Due to limited time constraints in the course, each project team is restricted to using this option only once in the semester and the failed deliverable must have originally received a grade of 69 or lower. To qualify for this credit, the entire team must make an appointment with the instructor to discuss the content and timing of the presentation.

Individual Portion

Oral Pattern Presentation

You are required to present a 10-minute in-class presentation on a design pattern or software process assigned by the instructor. Your presentation must be presented to the entire class. Some form of visual supplements are required, however slides are not mandatory. All presentations must include examples of **original** source code complemented by UML and a description of best practices used. Also include the benefits/drawbacks to the pattern(s).

A portion of this grade will depend on how well you use available media. Please keep in mind that available media is anything you can use in class: overhead transparencies, paper handouts, computer-based slides, websites, etc. If you are offering external resources (such as a website) please be prepared to show it in class as part of your presentation. All references must be cited in the proper form.

Once you deliver your presentation, any materials you present must be sent to the instructor for posting on the class website for your peers to review. Grades will not be provided until the presentation materials are received by Team Instructor.

If you set up an additional presentation with the instructor you can receive an additional 10 points to your presentation grade. This would be for a short (5 minute) presentation on some technology you and your team have adopted in your project this semester. This is completely contingent on the class schedule and needs to be coordinated with the instructor.

Take Home Exam

A single comprehensive exam will be distributed near the beginning of the semester. This exam is to be completed by the end of the semester as a final. The exam is open book and open notes. You are allowed to use any printed (or electronic) resource at your disposal except otherwise noted on the exam (e.g., Wikipedia). However, since the exam is an assessment of individual performance you are not permitted to share your answers with any other students, or to assist any other students with their exam. Please keep in mind that any resources used must be referenced as per standard academic practice. You are required to have a review of your draft exam by the Center for Communication Practices (http://www.ccp.rpi.edu/online.html). A bonus of seven (7) points is awarded to each student that receives a **second** review of their exam. (This bonus is only for students who hand in their exam on time.)

Every student must pass the exam in order to pass the course.

Peer Evaluation

At the start of the semester your project group will develop a set of metrics which the team members will use to evaluate each other's performance at the end of the semester. This evaluation of your performance and work ethic during the project will count for 15% of your individual grade.

Peer reviews are a very common way to evaluate employee performance in business; consequently the instructor takes the peer evaluations very seriously.

The peer metrics you devise should be scored on a scale from 1 to 25 for each of **four** specific metrics. You must submit an evaluation of your teammates in order to receive credit for this portion of the course.

Group Project

This course focuses on object oriented software development, and therefore the group projects must be implemented in an object oriented programming language, in an object-oriented way. With the number of mainstream OO languages available today, we don't require the implementation to be in any specific language. However, we ask that you receive approval from the instructor before beginning the implementation in any language which is not listed here:

- Smalltalk
- Java
- C#
- C++
- Objective-C
- Swift
- Python
- Perl
- PHP (v5 or later)
- VB.NET
- Ruby on Rails
- ActionScript
- Go

In all of these languages we require that you use the object-oriented features of the language in order to receive credit for the project. Be forewarned that failure to do so can have significant penalties on the group grade.

Selection of implementation technology must be documented in the project's artifacts. Please make sure you document and justify your technology selection(s). (For example, using PHP, a language which is primarily used to create dynamic websites, to build a remote controlled helicopter's software system might not be the best design decision.)

All team members are required to contribute to all portions of the group's deliverables. It is not acceptable to have one team member do the UML and documentation, another provide the implementation in the selected language, another do the requirements analysis, etc. The group project is designed to give you the opportunity to experience some development team skills before graduation so you are prepared for the realities of modern software development.

Based on the Contribution Summary submitted for each deliverable, it is important that each team member’s contribution is accurate. This is how each team reports the level and quality of each team member’s input to the project as a whole. The instructors reserve the right to penalize the grade of specific members if their contribution is substantially less than the other members of the team.

Project teams are required to select their own projects, design feature requirements, etc.

Therefore it is permissible to use an extended project from another (concurrent) course for the group project in this course. Please get approval from the instructor prior to beginning however, to ensure that the project has enough scope to justify two grades for one project (a grade in this course and a grade in the concurrently running course). Extending a project you did for a previous course is also an option, but please seek instructor approval if you intend to reuse any source code or documentation from the prior class project.

Project ideas can be obtained from the instructor if your project team is having a difficult time coming up with a project definition.

Project teams will be determined by the instructor at the start of the course. The project teams will be formed of 4 students from your section of SD&D, Some exceptions may be granted allowing a group to have 5 members.

All projects need to be approved by the instructor.

Assignment Submissions and Other Communication

Given the size and amount of material the instructor receives over the course of the semester, we request that you adhere to the following guidelines. Group project grades and presentation grades may be penalized for blatant disregard of these guidelines.

All group assignments are to be submitted via LMS (https://lms.rpi.edu/). If the submission requires multiple files, please zip them into a single file. If the deliverable is in a single file, just submit that file. Entitle your file with the name of your group, the name of the assignment, and the submission date. For example, the group WeRock should submit their Elaboration deliverables on November 5th by pushing the following file to the LMS site: WeRock-Elaboration-Nov11.zip or WeRock-Elaboration-Nov11.pdf.

All communication with the instructor should be sent to sddteaminstructor@gmail.com. We strongly recommend that you CC any communication which is about the team or the team project to everyone in the group. We welcome any and all communication with us through email.

Cheating

Simply put, DON'T CHEAT. If you are caught cheating on a test, you will receive a 0 for that test. Teams are expected to work independently of each other. If teams are caught cheating they will be referred to Dean of Students Office for disciplinary action.

Late Submissions / Missed Project Milestones / Low Project Grades

Late submissions are not acceptable, and significant points will be deducted for each day they are late. Submissions must be received in LMS by 11:59:59 pm on the day they are due, in the correct submission format(s). Failure to use the proper submission procedures may result in point deductions.

Instructor Office Hours

Office hours can be made by appointment via email (sddteaminstructor@gmail.com) for immediately before or immediately after class. Other times may be arranged, subject to instructor schedules. Please keep in mind that the instructor and TAs are not always available to meet with project teams on short notice! We prefer to meet with the entire project team whenever possible, unless the topic(s) of discussion are entirely individual in nature.