CSSE376: Software Quality Assurance

Instructor: Mark Hays

Office: Moench Hall, Room D205

Email: hays@rose-hulman.edu

Office Hours:

T 1:00 – 3:00
WF 2:00 – 4:00

Course Prerequisite: CSSE 230 **(**Data Structures and Algorithm Analysis) or equivalent

Course Description: CSSE 376 covers theory and practice of determining whether a product conforms to its specification and intended use. Topics include:

1. Software quality assurance methods;
2. Test plans and strategies;
3. Unit level and system level testing;
4. Software reliability;
5. Peer review methods;
6. Configuration control responsibilities in quality assurance.

Course Outcomes: Students who complete this course will be able to

1. Create a test plan for a software system.
2. Explain the differences between testing strategies (ie unit vs integration, white vs black box, etc).
3. Apply boundary value analysis to create a unit test plan.
4. Apply test adequacy measures (e.g. code coverage and mutation testing) to identify deficiencies in a project's testing.
5. Apply mocking to convert integration tests into unit tests.
6. Apply principles and strategies of integration and regression testing.
7. Explain purposes of metrics, quality processes, methods for measuring that quality, and standards used.
8. Conduct code reviews and formal code inspections.
9. Explain the software design choices that must be made when developing software for international audiences.
10. Apply principles of test driven development to successfully develop a software product.

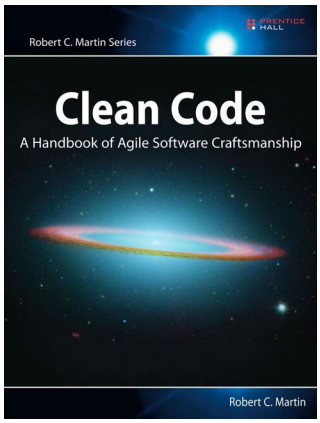
Required Textbook (yes, actually required)

Clean Code: A Handbook of Agile Software Craftsmanship

Robert C. Martin

Prentice Hall

978-0132350884



Learning Toolbox

Learn how to succeed in this course by following these tips.

What is the student's responsibility in the learning process?
Students from last year mentioned that the junior sequence was when they "learned how to learn on their own." We want to amplify that observation this year.

- Accept sole responsibility for your learning in this course.
- Actively read the assigned text before class.
- Start early: Make an initial attempt to start your assignments in-class the day they are assigned. Get a sense for the difficulty and the concepts involved.
- Ask questions early: In industry, asking questions during software engineering activities is a sign of experience, not weakness. Berenbach applauds the "smart ignoramus" for asking questions that other people assume have an obvious answer. Feel free to ask the instructor or TA about any questions you have. Bring your marked-up textbook and your reading notes.
- Take notes during group discussion and lecture: anything discussed in class is fair game for the exam.

How would an experienced learner read a text in your course?
We won't always have assigned reading, but when we do, expect each day's reading to take you three to six hours. This is consistent with the old rule of thumb that an hour of class requires three hours of study. Here are some tips to succeed with your reading assignments:

- **Spread it out**: the reading should take multiple sessions. You can't remember it all in one sitting.
- **Preview**: Skim once to read the headings. Read the discussion questions. Write the questions down at the top of your notes.
- **Read Actively**: Fully read each section. Take notes on what is being discussed in that section.
- **Create Questions**: Invent your own study questions as you read. Write them down as you go.

- **Review and re-read**: Write down the big-picture idea for the chapter. Try to answer your study questions solely from memory - when you reach this point, you "did the reading." Otherwise, go back and re-read what you missed.
- **Discuss**: Discuss your answers to the reading quiz questions with a study group. Come prepared for class discussion.

Do you encourage study groups?
Yes! Both for daily readings and for exam preparation. Complete your reading of the material and write your study questions before meeting with your study group. Test each other by swapping the study questions you each wrote as part of the reading. If you go to your study group unprepared, you could hurt the other students by posing inadequate questions.

How should students prepare for the exam?

- Re-read the notes you took in class and on the reading assignments.
- Review the reading quizzes. Reflect on why you selected your answers. Also reflect on the significance of the questions in forming your understanding of the reading.
- Test your peers by asking them the study questions that you created from the reading.

How should students prepare for the project?

- At every step of the way, set clearly defined goals.
- Split the work up so your team is working on several tasks at the same time.
- Read about the GitLab workflow assigned in the first week of class and adopt it.

Grade Division

To earn a passing grade in the course, you must have at least a D average in each of these categories:

Items	Weight
Class Participation	10%
Tutorials	10%
Exams	50%
Final Project	30%

Yes, this means you can fail the course by failing the final project grade at the very end of the term. This **rarely** happens, but when it does, the effect is usually limited to one team member who slacked off.

Class Participation

There are 40 meeting times during the term. Some days there will be quizzes, some days there will be some activity that will be graded, some days there will be project work time. However, some part of your overall participating will be my assessment that you participate in class discussions even outside of individual quizzes/activities.

Up to 2 unexcused absences are allowed. Any additional unexcused absences shall result in you receiving a failing grade for the course. You are solely responsible for making up any missed work.

Exams

We will take three exams. The first two exams will be split over the Thursdays and Fridays of 4th and 7th week. The third exam will be split over Tuesday and Thursday of 10th week. They will be in-class, paper-and-pencil, closed laptop, open book, and open notes. Answers will be graded **pass/fail**, with the bar for passing being, "Is there something you could learn by studying this material some more?"

You will have opportunities to grade-replace missed questions on subsequent exams. So for example, if get a question wrong on Exam 1, you can try again on the similar question on Exam 2. If you get that question right, you can leave that similar question blank on Exam 3 and it won't count against you. Questions will be numbered by course outcome and will align with the numbering in the Moodle gradebook so you will always know which questions you will need to re-attempt.

Accessibility

Rose-Hulman Institute of Technology strives to make all learning experiences accessible to students. If you anticipate or experience academic barriers based on accessibility issues, please feel free to communicate those needs and register with Student Accessibility Services. Student Accessibility Services will work with you understand the process and to determine what accommodations are most appropriate for your individual situation. Please note that accommodations are not retroactive and accommodations cannot be provided until verified. Please contact Student Accessibility Services for more information at HMU 156, phone 812-877-8040, or email eaton1@rose-hulman.edu. Please note that it is the student's responsibility to request any approved, documented academic accommodations (such as extra time) **at least three days in advance** of exams.

Tutorials

Tutorial exercises for this class will be used to expose you to various quality assurance practices that will be pertinent to the final project. They will always offer step-by-step instructions to guide you through the process of learning the technique at hand. Tutorials, unless stated otherwise, will be done individually.

There will be approximately 7 tutorials. The tutorials will likely comprise of the following:

1. Git and merge conflicts

2. Test Driven Development
3. Code Review
4. Code Coverage
5. Behavior Driven Development
6. Performance Profiling
7. Automated GUI testing

Grader feedback and retries
Tutorials will be graded pass/fail. If you earnestly complete the assignment but make some error along the way, your grader will assign a fail grade, but will leave feedback asking you to submit a correction.

So when you see a failing grade on a tutorial, don't panic! Read the feedback, make the requested corrections, and notify the grader of your changes to get your grade changed.

Late Policy
Please note that tutorials unless otherwise specified will be due at 5:30 PM on the Friday of the same week of the tutorial. This course has no 'late day' policy and late assignments are generally worth 0 points unless you have made special arrangements.

However we understand that situations can arise. We handle all situations on a case-by-case basis. Here are some general guidelines:

- We expect to be notified as soon as is feasible. If you are very ill, we understand that that might be slightly after the assignment due date. For other situations, it's usually possible to notify us in advance.
- The usual extension we will provide is 24 hours. Occasionally we may allow 48 hours. More than that is very unlikely.
- We keep track of extensions per student and if you seem to be overusing the privilege we will stop allowing extensions.
- If the reason you are late is because you are having trouble completing the assignment (e.g. a bug you can't fix or just difficulty understanding) get help from an instructor. Don't assume that more time will fix the problem; plan to get help.

Final Project

Students will be working on an end of the term project during the class. We will be using Test Driven Development as the model for software development. According to Wikipedia, "Test-driven development (TDD) is a software development technique that uses short development iterations based on pre-written test cases that define desired improvements or new functions. Each iteration produces code necessary to pass that iteration's tests. Finally, the programmer or team refactors the code to accommodate changes. A key TDD concept is that preparing tests before coding facilitates rapid feedback changes. Note that test-driven development is a software design method, not merely a method of testing."

The software application that you developed should also be localized. Localization refers to the process of translating a product into different languages or adapting a language for a

specific country or region. We expect your software application to support two languages at the very least.

Team Composition
This is a big project, so teams will consist of about 3-4 members each. We are willing to allow smaller teams, if you have a good reason for wanting to do so. We will use CATME to assign teams.

Project Selection
Software engineering is a creative endeavor and can be a lot of fun. We want you to choose a project that is interesting to you. We impose some constraints below so that the project will also let you demonstrate your skills relative to the learning objectives of the course. Each team will initially have a project idea, based on discussion within the team. Your team's first task is to work with the instructor and assistants to refine that idea so that it satisfies the following constraints:

1) It should have a valid need and the solution should have at least 6-7 features.
2) The requirements must be valid, measurable, testable, stable and feasible.
3) It should require 100 - 150 person hours to complete. That's about 4-5 hours coding per week per person – but bear in my that everyone codes at a different speed and so it might be a little more or less for you personally.
4) Please focus on projects that have a lot of business logic. Complex board games Settlers of Catan, Risk etc. usually work well. You are welcome to propose any project idea that has a sizable logic component. Based on prior experience, we request you to stay away from mobile applications, as students struggle to see past the newness of mobile to even think about unit-testing properly.
5) It must have a user-interface. A very basic user interface is sufficient for full credit.

Project Grading
You will receive feedback in various forms all throughout the term. But you will receive only **one** project grade at the end of the term. The grade will be based on a project rubric on Moodle and your individual contributions to the team. The rubric will be posted at the start of the term on Moodle at the top of the page. It is designed to be unambiguous to help you gauge your progress for yourself. To be clear though: **if your personal contributions to the project do not keep pace with your team, you will not earn the project's grade and can outright fail the course** on this basis alone. The project is a key component of our assessment of the more advanced learning outcomes, so if you don't put in the effort to demonstrate that learning, then you can't pass the course.

General Writing Issues
Written communication is important in this course, as it is in the profession in general. Remember that a software document has several unique and important characteristics:

1. Technical documents are often the result of group authorship, thus it requires planning and final tweaking.
2. Specificity and organization are more important than flow, hence technical documentation is often ordered around lists and tables rather than paragraphs.

3. Documentation is often the reader's only source of information on the particular subject or product, hence it must be thorough and complete.
4. Documentation is often used to answer a specific question; hence it should facilitate finding a specific piece of information (navigation).
5. Documentation must bridge from general specifications to particulars of implementation and operation, hence it must make abstract concepts concrete and make concrete facts fit generalized concepts.

You can always drop by my office or consult with one of the teaching assistants, if you have any questions regarding your document. We would be happy to look at it and suggest some changes. You should also be aware of the service provided by the Learning Center.

Collaboration

An outcome of this class is to collaborate in your project work, including doing this in new ways. Very much something we want!

You are encouraged to discuss the homework and other parts of the class with other students. Such discussions about ideas are not cheating, whereas the exchange of code or written answers is cheating. However, in such discussions of ideas, you should distinguish between helping and hurting yourself and the other student. In brief, you can help the other student by teaching them, and you can hurt them by giving them answers that they should have worked out for themselves. The same applies to tutoring and getting help from the instructor.

If you use reference materials (other than the course texts) to solve a problem, please cite the referenced material using IEEE style. Similarly, if you discuss a solution with another student, give credit where credit is due by making a note such as "the following idea was developed jointly with Alyssia P. Hacker," or "the following idea is due to Ben Bittwiddle." You cannot be charged with plagiarism if you cite in this way. (However, don't expect to excuse cheating with such a citation. That is, you cannot exchange code even if you say it was developed in cooperation with someone else. Cooperation refers to the exchange of ideas, not code or written answers.)

The instructors take plagiarism seriously. Students caught plagiarizing, or cheating in any way, will be reported to the dean of students and will receive an automatic F in the class.

Self-plagiarism

Students who turn in work derived from a past attempt at the course shall receive an F in the course.

Self-plagiarism is copying and reusing all or some of your previous work for another course attempt. There are several issues with reusing your own work, including:

1. It probably wasn't of high quality (since you're here retaking the course).
2. It gives you a time advantage over other students.

3. You are not learning the material by resubmitting existing work.

While all of these are significant issues, the last one is our biggest concern. If you are retaking the class, you should be redoing the work, as the practice helps reinforce course concepts.

Self-plagiarism will be treated as any other form of academic misconduct.

Generative Artificial Intelligence

The CSSE department is adopting a department-wide ban on the use of generative AI in courses, such as this one, where your coding skill is a focus of the course learning outcomes. The sections below come from our joint efforts to illustrate that policy.

In summary: work for this course must be your own! This means that you wrote or designed it on your own, **without** the use of generative AI platforms. **Work wholly or partially created through generative AI will be treated as any other form of academic misconduct.**

What is considered generative AI?

The term generative AI is not necessarily restricted to large language models accessible through online platforms, such as ChatGPT or Bard. Plugins that use generative AI to complete code fragments (mainly GitHub Copilot) also fall under this policy and will be treated similarly. Using any plugin in your IDE that interacts with these generative AI models will be considered as academic misconduct.

Why no generative AI?

We want you to learn how to do it. Yes, generative AI will be a part of our future, but it may not always be available to you, and if you do have access, you need to understand if the work you receive from it is any good. In the first case, generative AI platforms can store and repeat what they're given. Imagine working with confidential company information or in classified government work? You cannot feed it the information on what you want, because that information will become part of its network. Beyond that, you cannot simply assume that an AI did good work. Even if you end up in a job that is AI-assisted, a human still needs to be able to evaluate if the work was done properly or not, you won't be able to do that unless you have experience doing the work yourself.

Syllabus Changes

The syllabus is a living document. We reserve the right to change the syllabus at any time, but you will be alerted to any changes.

Final Grade

The Institute recognizes the course instructor as the sole authority in the matter of determining final grades. As such, the course instructor reserves the right to enter any final grade that the instructor believes you earned, even if that grade contradicts other

information, such as Moodle, previous statements by the instructor, or any statement in this syllabus.

Syllabus developed by Mark Hays for Spring 2024, based on earlier work by Mark Ardis, Donald Bagert, Victoria Bowman, Mike Hewner, and Sriram Mohan.