CEG 3110/5110: Introduction to Software Testing

Fall Quarter, 2015

Course Description

This course covers formal software testing strategies, along with established best practices, so students learn how to test their software in a complete and systematic (vice ad-hoc) manner. Particular attention is paid to planning, writing, and executing software tests, along with associated documentation, (i.e., a software test plan), which includes documented results. Various projects are assigned, designed to illustrate various challenges associated with software testing, and to reinforce the strategies and techniques used to overcome these challenges.

Location & Time

This class is designated to meet in **Fawcett 204**, on Mondays and Wednesdays from 6:10 to 7:30.

Textbook

Paul C. Jorgensen, *Software Testing: A Craftsman's Approach*, This is a required textbook for this course. You can obtain either the third or the fourth edition of the book for this course.

Reading Assignments

Each week's lessons have corresponding reading assignments. The course lectures are designed to *augment* (not simply *rehash*) these readings. Students should not let their textbook collect dust over the course of the semester.

The course text is a straightforward book. Chapters are written succinctly. It would behoove students to review the material in the book during the week when it is being covered in class.

Course Projects & Lectures

This will be a "learning-by-doing" class. Students will have a series of projects throughout the course, where they will write code, write test plans, execute test plans, and document the results. The course projects represent rather unique programming assignments, in that grading is based primarily on how well the code is ***tested***, as opposed to how well it is written.

Class time will consist of interactive discussions. Students should attend class ready to contribute through active participation. **During class, you should be engaged in the discussion, not multitasking on your laptop, tablet, or cell phone.** Students who take notes on iPads, etc., can take notes on their tablet, but, please, **no** checking social media sites under the pretenses of "taking notes." Class time should be devoted to learning, not browsing.

Instructor Contact Info

John Reisner
Office hours after class or by appointment
Daytime Phone: 255-3636 x7422 (this is a WPAFB phone number).
email: **john.reisner@wright.edu** (for a more timely response, consider sending a CC: to **jreisner@afit.edu**).

The instructor is an adjunct faculty member. Most contact will be done via email, phone, or during before- or after-class discussions. Other meetings can be arranged as needed.

Course Objectives

Each student should be able to:

1. Write well-organized and appropriately comprehensive test plans using good writing techniques.
2. Effectively document test plans and results.
3. Develop software using a test-driven approach.
4. Employ effective testing strategies for different needs.
5. Write drivers, stubs, and testware as needed to sufficiently test a program.
6. Verify a program's correctness via a test strategy.

Course Components

Being a higher-level college course, my goal is to teach at the higher levels of **Bloom's Taxonomy**. Overall, the goal of this course's homework assignments, projects, and exams are to stimulate and promote ***thought***, particularly at higher levels of the cognitive and affective domains. Effective writing applicable to real-world software engineering is also strongly emphasized.

35% Course Projects

- There will be 3 or 4 course projects; each will involve programming. However, the emphasis of these projects will be **testing** the software that has been tested, *with a written test plan* that has been developed by the student. (This is primarily a *testing* course, not a *programming* course.)
- Each project will have separate phases where (a) requirements are written, (b) test plans are developed, (c) code is written and debugged, and (d) test plans are run with results recorded.
- Each project will be graded individually. Although the grade will be based primarily on the thoroughness and quality of the test plan, students are expected to use good programming style and employ accepted programming practices throughout the course. Documentation must also employ good grammar and writing style. Test plans need to be well-organized and written such that an outside tester could use the test plan to test the software.
- Projects increase in complexity over the duration of the course. The final project will span over several weeks and require significant effort.
- The student may select the programming language used for each project.
- Projects are to be submitted in paper form, to include code listings.

20% Midterm Exam

- Mixed-format exam, administered in class. Exams usually consist of seven multiple-choice questions (28 points), plus somewhere between four and eight short answer, essay, or analysis questions (72 points).
- Any of the material covered in assigned readings, or in-class discussions is considered "testable." That said, **tests are designed to focus on and reiterate important and fundamental concepts, rather than minutiae**.
- Many exam questions are designed to test the ability of the student to analyze, synthesize, and eloquently explain information and concepts, rather than recall simple facts. When you are asked to argue a certain position, your argument is expected to be sound, and it should clearly demonstrate that you are learning in the course.

25% Final Exam

- Comprehensive, mixed-format exam, administered during the school's final exam week. This exam will resemble the midterm (although the questions will be different, of course).

15% Homework Assignments

- There are two kinds of homework assignments. (1) Some homework assignments are milestones for the course project; this is done to prevent students from procrastinating, and also to provided feedback that can be incorporated into a student's final project. Other homework assignments may differ from the course projects; these assignments are designed to facilitate deeper comprehension about a lecture topic.
- In general, homework will be due on the Monday following the day it was assigned; therefore, students will have one week to complete homework assigned on a Monday, and five days to complete homework assigned on a Wednesday.
- Deadlines and details about each assignment will be announced in class and posted on Pilot.

5% Class Participation

- Based on attendance, attentiveness, attitude, participation, readiness to learn, and willingness to share thoughts and ideas.

Grading of Course Work

All homework assignments and projects must be submitted in paper form.

At the end of the term, the following scale is used:

92-100 = A
84-91 = B
75-83 = C
65-74 = D
< 65 = F

However, **this scale may be (and frequently is) curved**, at the instructor's discretion, after all work has been graded, and the grade distributions have been analyzed.

Many of the assignments in this class will be graded subjectively, due to the nature of the work. Many assignments require turn-ins that are not necessarily *right* or *wrong*, but rather well- or poorly-documented, strongly or weakly substantiated, thorough or cursory, pithy or superficial, well-organized or carelessly compiled. Superior work is graded above 90; satisfactory work is graded between 80 and 90, and unsatisfactory work is graded below 80, depending upon the severity of the problems. Grading rubrics will be employed from time to time, to help students gain a better understanding of expectations.

Late Work

- No late work will be accepted after the last day of class for the quarter.
- Before the last day of class, late work is accepted, with some points deducted based on how late the assignment is turned in. (The later the work is turned in, the more substantial the deduction). Extenuating circumstances will be considered; advanced notice (via email) is considered a good practice.
- If you complete a late assignment several days before we meet for a class (e.g., say you complete an assignment on a Thursday; but we don't meet again until Monday), it is generally best to email me the work right away (this email will stop the "lateness penalty clock"). However, you must *also* print a copy, and hand in that hardcopy during the next class session (which prevents me from assessing the "instructor had to print this" penalty).

IW Credit

This course can be used for IW credit. Because of that, I must mention: during this course, Wright State students will be able to produce writing that

- Demonstrates their understanding of course content,
- Is appropriate for the audience and purpose of a particular writing task,
- Demonstrates the degree of mastery of disciplinary writing conventions appropriate to the course (including documentation conventions), and
- Shows competency in standard edited American English.

Because of this, it is imperative that your test plans are clear, thorough yet concise, well-organized, and unambiguous. It is not enough to simply have the right test cases – they must be presented in a manner that would facilitate testing for whoever would be running the test plan and reassure the program manager that comprehensive testing has been performed.

Overall writing quality **will** be assessed and be part of your grade for all projects.

Course Schedule (subject to change)

Week	Lsn	Date	Lesson Topics	**Assigned Reading** (3rd/4th)
Part 1 - Course Introduction				
1	**1**	**Mon Aug 31**	**Course Introduction** **Terminology & Basics**	Chapters 1 & 2
	2	**Wed Sep 2**	**The V-Model & Testing**	
Part 2 - Functional (Black Box) Testing				
2		**Mon Sep 7**	**LABOR DAY**	
	3	**Wed Sep 11**	**Equivalence Partitioning**	Chapter 14
3	**4**	**Mon Sep 14**	**Boundary Value Testing**	Section I Introduction Chapters 3 & 4
	5	**Wed Sep 16**	**Orthogonal Arrays**	Chapter 6
4	**6**	**Mon Sep 21**	**Decision Tables**	Outside Readings
	7	**Wed Sep 23**	**Stress, Performance, and Load Testing**	Chapters 5 & 7
5	**8**	**Mon Sep 28**	**Designed-Based OO System Testing**	Chapter 8
	9	**Wed Sep 30**	**Use-Case Testing**	Chapter 12/11
6	**10**	**Mon Oct 5**	**MIDTERM EXAM**	
Part 3 - Test-Driven Development				
6	**11**	**Wed Oct 7**	**Intro to Test-Driven Development**	Chapter 23/19
7		**Mon Oct 12**	**Review Midterm**	
	12	**Wed Oct 14**	**Exploratory Testing**	Section III Introduction Chapter 13
8	**13**	**Mon Oct 19**	**TDD vs Conventional Coding**	
	14	**Wed Oct 21**	**Project 2 Prep**	
9	**15**	**Mon Oct 26**	**Project 2 In-Class - *bring your laptop!***	
Part 4 - Structural (White Box) Testing & Other Testing Topics				
9	**16**	**Wed Oct 28**	**White Box Testing**	Section II Introduction
10	**17**	**Mon Nov 2**	**White Box Testing Coverages**	Chapter 11/10
	18	**Wed Nov 4**	**Exploratory Testing**	
11	**19**	**Mon Nov 9**	**Regression Testing**	
		Wed Nov 11	**VETERAN'S DAY**	
12	**20**	**Mon Nov 16**	**Software Metrics**	Chapter 16/15
	21	**Wed Nov 18**	**Testing Metrics**	Chapter 25/23
Part 5 - Course Conclusion				
13	**22**	**Mon Nov 23**	**In-Class Q-&-A *bring your questions!***	
	23	**Wed Nov 25**	**HAPPY THANKSGIVING**	
14	**23**	**Mon Nov 30**	**"Stop! In the Name of Love"**	
	24	**Wed Dec 2**	**Final Project Q-&-A**	
15	**25**	**Mon Dec 7**	**Sundry Topics**	
	26	**Mon Dec 9**	**Course Review**	

Appendix A: Comparison between 3rd & 4th editions of course text

			bold	both eds.
Edition			*ital*	3rd ed only
3rd ed.	4th ed.	Chapter Title	plain	4th ed only
1	1	**Perspectives**		
2	2	**Examples**		
3	3	**Discrete Math**		
4	4	**Graph Theory**		
5	5	**Boundary Values**		
6	6	**Equivalence Classes**		
7	7	**Decision Tables**		
8		*Functional Testing Review*		
9	8	**Path**		
10	9	**Dataflow**		
11		*Structural Testing Review*		
	10	Unit Testing Review		
12	11	**Levels of (Lifecycle) Testing**		
	12	Model-Based Testing		
13	13	**Integration Testing**		
14	14	**System Testing**		
15		*Interaction Testing*		
16	15	**OO Testing**		
	16	Complexity/Metrics		
17		Class Testing		
	17	Systems of Systems Testing		
18		*OO Integration Testing*		
19		*GUI Testing*		
20		*OO System Testing*		
21	18	**Exploratory Testing**		
22		*Model-Based Testing*		
23	19	**TDD**		
24	20	**Pairs Testing - Closer Look**		
	21	Evaluating Test Cases		
	22	Software Technical Review		
25	23	**Epilogue: Testing Excellence**		