COMP 587: Software Verification and Validation
Spring 2020

Instructor: Kyle Dewey (kyle.dewey@csun.edu)
Course Web Page: https://kyledewey.github.io/comp587-spring20
Office: JD 4419, Extension 4316 (not yet connected)

Course Description (From the Catalog)
An-in depth study of verification and validation strategies and techniques as they apply to the development of quality software. Topics include test planning and management, testing tools, technical reviews, formal methods and the economics of software testing. The relationship of testing to other quality assurance activities as well as the integration of verification and validation into the overall software development process are also discussed.

Learning Objectives
Successful students will be able to:
- Formulate an actionable test plan with ACC (Attributes, Components, Capabilities)
- Apply their test plan to testing existing software with industry-standard techniques
- Understand and apply advanced testing techniques to existing software
- Use model checking techniques to build confidence in the correctness of software specifications
- Use verification techniques to prove small programs correct

Course Motivation (or a Relevant Rant)
As a society, we have chosen to surround ourselves with software, and to make software essential to daily life. Software controls everything from a car's fuel injector to critical infrastructure. Software makes it easy to build new things cheaply, and to rapidly adapt to change.

However, as a society, we have counterintuitively decided that it's generally *ok* for software to fail, especially if failures are rare. It's ok when unanticipated inputs cause an app to crash. It's ok if memory leaks require me to power cycle a router. It's ok if a race condition causes the traffic lights at an intersection to go to flashing reds. It's ok if a flaw in my encryption scheme allows others to read my email. It's ok if an unchecked array access leaks my social security number. It's ok if a missing bounds check destroys my life's work. It's ok if a lack of input sanitization wastes hundreds of millions of dollars. It's ok if an unexpected scenario kills people.

This sounds ridiculous, and it is ridiculous, but this is reality. Each of the above examples is rooted in real software flaws with real impacts. None of these examples have resulted in widespread outcry for better software. If you think any of the above are decidedly *not ok*, and you want to do something about it, this class is for you.

Textbook
No textbooks are required. That said, the following books may be of interest to you:

- How Google Tests Software, James Whittaker, Jason Arbon, Jeff Carollo - discusses ACC and a number of basic industrial testing approaches
- Fuzzing: Brute Force Vulnerability Discovery, Michael Sutton, Adam Greene, Pedram Amini - discusses some of the advanced approaches; fairly out of date, but many of the concepts are the same
- Specifying Systems: The TLA+ Language and Tools for Hardware and Software Engineers, Leslie Lamport - introduction to formalizing and reasoning about software specifications with the TLA+ model checker, by the creator of TLA+
- Software Abstractions: Logic, Language, and Analysis, Daniel Jackson - similar to the book before, but for the Alloy model checker by the creator of Alloy
- Certified Programming with Dependent Types, Adam Chlipala - an in-depth look at dependent type theory and Coq, suitable for proving software implementations correct

Graded Components
Your grade is based on the following components:

Assignments	40%
Project - Proposal and Formulating Test Plan with ACC	4%
Project - Unit Testing with Sufficient Coverage	12%
Project - Automated Testing	15%
Project - Pull Requests	5%
Project - Final Report	4%
Midterm Exam	10%
Final Exam	10%

Not all assignments will be weighted evenly, nor will you always be given the same amount of time for assignments. Exactly which assignments are assigned depends on how the class progresses. In general, assignments will be submitted through Canvas (https://canvas.csun.edu/). In the event that there is a problem with Canvas, you may email your assignment to me (kyle.dewey@csun.edu), though this should be considered a last resort.

Class content is reinforced by assignments (40% of your grade), which broadly cover V&V. The class project (40% of your grade) further requires you to take some of the lessons you've learned from the assignments and apply them to some preexisting piece of software. The assignments are intended to give you a (hopefully) gentle introduction to a variety of techniques, whereas the project lets you apply some of these techniques in a more realistic setting. Details of the project follow, roughly in chronological order:

1. System under test (SUT) selection. You will choose a software project which you want to apply V&V techniques to, hereafter referred to as a SUT. The SUT should be sufficiently large (at least 1,000 lines of code for each person on the project), and can be in any state of development. Your grade is based **solely** on the V&V aspect, so if you select an incomplete SUT (e.g., it is still in the planning stages), the actual SUT implementation is secondary. You are encouraged to select software you have written in the past, but you must be ok with giving everyone in the class access to this code. You may work individually if you so choose.
2. Proposal and formulating a test plan with ACC (4%). This describes what exactly you'll do in the project.
3. Unit testing with sufficient coverage (12%). You'll apply unit testing to your project to attempt to find bugs and improve confidence. To ensure your unit testing is sufficiently, you'll measure its effectiveness using code coverage measurement techniques.
4. Automated testing (15%). You'll apply automated testing techniques to some part of your project. For our purposes, "automated testing" means "automatically generating test inputs and automatically running them".
5. Project pull requests (5% total). You are required to submit 5 pull requests, each worth 1% apiece, roughly one every two weeks you work on the project. Each pull request consists of a separate contribution to the project. These are intended to get you familiar with filing pull requests, and to ensure that steady progress is made on the project. **You cannot submit them all at once.**
6. Project report (4%). You will write background information on the project, how you tested it, and what you've learned in the process.

The exams are intentionally not worth too much. Most of what we learn in the class is necessarily heavy on programming with an emphasis on interaction with the SUT; most of this is not feasible to do in an exam environment without a computer. As such, the exams focus on higher-level, more conceptual issues.

Final Grades
Plus/minus grading is used, according to the scale below:

If your score is >=...	...you will receive...
92.5	A
89.5	A-
86.5	B+
82.5	B

If your score is >=...	...you will receive...
79.5	B-
76.5	C+
72.5	C
69.5	C-
66.5	D+
62.5	D
59.5	D-
0	F

Plagiarism and Academic Honesty
While collaboration is allowed on assignments, you are responsible for all of your own work. You may **not** take code from online sources and submit it as your own. If you do find code online which you wish to include in a solution, you must **cite it**. Any violations can result in a failing grade for the assignment, or potentially failing the course for egregious cases. A report will also be made to the Dean of Academic Affairs. Students who repeatedly violate this policy across multiple courses may be suspended or even expelled.

Attendance
I will take attendance for the first two class sessions. If you miss both sessions and have not made alternative arrangements with me, you must drop the class, as per University policy (http://catalog.csun.edu/policies/attendance-class-attendance/). This policy is in place to help motivated waitlisted students enroll in the class. After the first week I will not take attendance, though you are strongly encouraged to attend.

Communication
- Canvas' messaging is encouraged for anything that is potentially relevant to the whole class
- Email is preferred for anything specifically for me

Late Policy / Exam Scheduling
Late assignments will be accepted without penalty if prior arrangements have been made or there is some sort of legitimate emergency (at my discretion). If you must be absent from an exam, contact me ASAP to see if alternative accommodations can be made.

If an assignment is otherwise submitted late, it will be penalized according to the following scale:

If your assignment is late by <= this many days...	...it will be deducted by...
1	10%
2	30%
3	60%
4+	100%

To be clear, assignments which are submitted four or more days beyond the deadline will not receive credit. The reason for such a harsh late policy is that we will generally discuss solutions in class shortly after the deadline, and this late policy discourages people from simply pulling a solution from an in-class discussion.

Class Feedback
I am open to any questions / comments / concerns / complaints you have about the class. If there is something relevant you want covered, I can push to make this happen. I operate off of your feedback, and no feedback tells me “everything is ok”.

---Class Schedule and List of Topics on Next Page---

Class Schedule and List of Topics (Subject to Change)

Week	Monday	Wednesday
1	~~1/20~~: MLK, Jr. Day (No class)	1/22: Introduction, motivation, project information
2	1/27: Project information, ACC-based test planning	1/29: ACC + project, git and GitHub
3	2/3: Linters, unit testing, testability	2/5: Unit testing, testability
4	2/10: Unit testing, testability, mocking	2/12: Continuous integration
5	2/17: Measuring testing effectiveness: test coverage	2/19: Measuring testing effectiveness: mutation analysis
6	2/24: Mutation analysis, automated testing introduction (basic concepts + loops)	2/26: Grammar-based automated testing
7	3/2: Grammar-based automated testing	3/4: Grammar-based automated testing
8	3/9: Grammar-based automated testing	3/11: Property-based testing: introduction + relationship
9	~~3/16~~: Spring Recess (no class)	~~3/18~~: Spring Recess (no class)
10	3/23: Lab (work on projects)	3/25: **Midterm Exam**
11	3/30: Property-based testing	4/1: Property-based testing
12	4/6: Concolic execution	4/8: Verification introduction, Dafny
13	4/13: Verification with Dafny	4/15: Verification with Dafny
14	4/20: Verification with Dafny	4/22: Model checking introduction, Alloy
15	4/27: Verification with Alloy	4/29: Verification with Alloy
16	5/4: Verification with Alloy	5/6: Final review