

CSE 4415 Software Testing 2 (3 credits)

Primary instructor: Keith Gallagher

Supporting instructor: Cem Kaner

Textbooks and references:

P. Jorgensen. Software Testing: A Craftsman's Approach, 4th edition. Auerbach Publications, 2008. (T)

Course information:

2014–2015 Catalog description: CSE 4415 Software Testing 2 (3 credits). Explores structural (glass box) methods for testing software. Includes testing of variables in simultaneous and sequential combinations, application programmer interfaces, protocols, design by contract, coverage analysis, testability, diagnostics, asserts and other methods to expose errors, regression test frameworks, test-first programming. Prerequisites: CSE 2410. Co-requisites: CSE 3101.

Prerequisites by topic: Java programming through data structures, software testing techniques at the black box level (ability to design tests, apply several techniques, evaluate results)

Place in program:

Computer Science Program: Advanced elective

Software Engineering Program: Required

Course outcomes & related student outcomes: The student will be able to

1. Relearn basic programming skills through a lens of writing code that always works — move from programming to software engineering. (4a: Skillful software construction)
2. Learn several glass box testing techniques. Develop skill in implementation-level verification of code. (2: Scientific, computing, and engineering problem solving)
3. Adopt industry standard development tools and use them effectively. (3: Skillful use of tools)
4. Improve cognitive skills in basic development, especially problem decomposition and technical communication (programmer to programmer). (7: Communicate effectively)
5. Become familiar with the agile approach to development lifecycles, and gain some experience with agile approaches, especially refactoring. (2: Scientific, computing, and engineering problem solving)
6. Work through exercises, and develop work products that help in interviews for new positions and lead to success in first year on the job. (9: Continually learn)

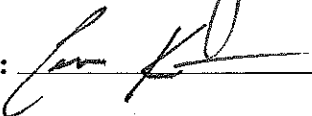
Topics covered:

1. Test-driven programming (3 hours)

2. Unit testing (5 hours)
3. Issues in the design of unit tests (such as testing Booleans) (3 hours)
4. Modeling the program as a state machine (3 hours)
5. Types of coverage (3 hours)
6. Working with source control (2 hours)
7. Working with an IDE (2 hours)
8. Using a source code analyzer to check style (2 hours)
9. Programming with a scripting language (Ruby) (6 hours)
10. Using Ruby to create API tests through the COM interface (3 hours)
11. Creating test tools that drive programs at the API level rather than the user-interface level (4 hours)
12. Maintaining a shipping open-source test tool (3 hours)

Approved by: Keith Gallagher, Associate Professor, Director of Software Engineering Programs & Cem Kaner, Professor

Signature:  Date: 2 FEB 15

Signature:  Date: 2/2/15