

## COURSE DESCRIPTION

**Department and Course Number:** COMP 6710

**Course Title:** Software Quality Assurance

**Total Credits:** 3

**Required:** Yes (SWEN)

**Prerequisites:** COMP 3700 or COMP 3710

**Class meetings per week:** 3 hours

**Lab meetings per week:** 0 hours

**Course Coordinator:** Dr. David Umphress

**Date Prepared:** February 18, 2004

### **Current Catalog Description:**

Processes, methods, and tools associated with the production of robust, high-quality software.

### **Textbooks:**

Pressman, R. 2005. *Software Engineering: A Practitioner's Approach, 6<sup>th</sup> Edition*. McGraw Hill. ISBN 0-07-285318-2.

### **References:**

1. JUnit – <http://www.junit.org>
2. McCabe and Associates – <http://www.mccabe.com>
3. Rational software from IBM – <http://www.rational.com>

### **Course Objectives:**

1. Understand the reason and purpose of software quality assurance.
2. Comprehend the ingredients of quality assurance: software quality assurance; configuration management; verification and validation; and test and evaluation.
3. Understand software life cycles and software processes.
4. Know the mechanics of Cleanroom software engineering.
5. Understand the functions of configuration management: identification, control, auditing, and status accounting.
6. Know the models of automated configuration management: check-out/check-in, composition, long transaction, and change tracking.
7. Understand the techniques of verification and validation: reviews, inspections, walkthroughs, and audits.
8. Understand graph theory as it applies to source code representation.
9. Be able to construct test cases for path, branch, and usage coverage.
10. Be able to construct test cases for equivalence class partitioning, boundary value analysis, cause-effect analysis.
11. Be able to construct a test strategy for unit, integration, and system tests.
12. Be able to articulate automated unit testing with JUnit.
13. Understand metrics common to software quality assurance.

### **Prerequisites by Topic:**

1. Software design
2. Familiarity with Java, C, or C++

**Topics Covered:** (specify number of hours on each)

1. Overview, verification and validation (2 hours)
2. Software quality assurance and engineering (3 hours)
3. Software process overview (3 hours)
4. Configuration management (2 hours)
5. Reviews and inspections (2 hours)
6. Testing overview (1 hour)
7. Graph theory (2 hours)
8. Structural testing (5 hours)
9. Functional testing (4 hours)
10. Testing strategies (4 hours)
11. Metrics (3 hours)
12. Selected quality assurance topics/presentations (11 hours)
13. Exams (3 hours)

**Laboratory Projects:** (specify number of weeks on each)

1. Inspections (1 week)
2. Structural testing (3 weeks)
3. Functional testing (3 weeks)
4. Regression testing with JUnit (2 weeks)

**Oral and Written Communications:**

Each student is responsible for a 15-20 minute presentation on a software quality assurance topic.

**Social and Ethical Issues:**

Although not a formal part of the course, social and ethical issues related to software quality assurance (e.g., safety and reliability) are discussed during appropriate lectures.

**Theoretical Content:**

Graph theory (2 hours)

**Problem Analysis and Solution Design:**

All students apply fundamental software engineering practices to analyze, design, implement, test, and document solutions to all programming assignments. Problem analysis and solution design are put in the context of verification and validation. That is, students are taught techniques for ensuring appropriate quality of analysis, design, and implementation efforts.